h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Wintersemester 2009/2010

Projektseminar

Praktikumsbericht

Navi3D

von

Christian Bartel, B.Sc.
Achim Brauer, B.Sc.
Tobias Braun, B.Sc.
Andreas Brust, B.Sc.
Antje Gloystein, B.Sc.
Sebastian Hohmann, B.Sc.
Thi Dieu Hien Le, B.Sc.
Christian Müller, B.Sc.
Dipl.-Inf.(FH) Marcus von Rohden
Julia Runge, B.Sc.
Johannes Seidel, B.Sc.
Alexander Steiger, B.Sc.
Andreas Völlger, B.Sc.

bei

Prof. Dr. Joachim Wietzke
Prof. Dr. Elke Hergenröther

Inhaltsverzeichnis

1 Einleitung

Ab diesen Semester steht ein neues Target mit einem Intel Atom-Prozesser und einem Nvidia Ion-Grafiksystem zur Verfügung. Auf dem neuen System soll ein Embedded Linux installiert werden, welches so minimal wie möglich gehalten werden soll, damit insbesondere die Startup-Phase kurz ist. Wegen des neuen Targets und eines neuen GPS Empfängers muss die existierende GPS-Komponente angepasst werden. Weiterhin muss ein neues Bedienteil (Minicommander) in Betrieb genommen und eine Framework-Komponente hierfür entwickelt werden.

Um dem Fahrer weitere Hilfestellung beim Autofahren zu geben, soll eine Fahrspurerkennung eingebaut werden. Diese soll dem Fahrer Informationen darüber geben, welche Fahrspur er benutzten muss um der Route zu folgen und auf welcher Fahrspur er sich befindet. Hierfür soll mit Hilfe einer im Auto montierten Kamera die Umgebung aufgenommen und die Fahrspur erkannt und klassifiziert werden.

Ein weiteres Ziel des Projekts ist die computergestützte Erkennung von Straßen-Verkehrsschildern. Für die Implementierung wurden teilweise Funktionen aus der Computer Vision Bibliothek „OpenCV“ verwendet. Der anfängliche Versuch die gesamte Implementierung ohne Verwendung von vorgefertigten Bibliotheken vorzunehmen, wurde nach einigen Wochen, aufgrund des Zeitdrucks und vor allem zur Reduktion der Rechenzeit, während des Betriebs vorerst verworfen. Ein weiterer großer Vorteil bei der Verwendung von OpenCV, ist die Unterstützung verschiedenster Bild- und Filmformate, wie beispielsweise jpg oder mpeg. Die Detektion wurde noch nicht mit dem vorhandenen Video-Material getestet, da die gefilmten Verkehrsschilder, aufgrund der schlechten Qualität des Materials, selbst für das menschliche Auge, kaum zu erkennen sind.

Das vorhandene und eingesetzte Framework war anfänglich nur für Windows und QNX, einem proprietären Echtzeitbetriebssystem, kompilierbar und lauffähig. Um eine höhere Flexibilität zu erreichen wurde ein Rechner mit Atom-Prozessor (sog. Nettop) angeschafft, auf dem Linux zum Einsatz kommen soll. Aus diesem Grund ist es erforderlich, das Framework auf Linux zu portieren und entsprechend lauffähig zu bekommen. Ziel ist der Einsatz des Frameworks auf diesem neuen Zielsystem.

Für das neue Target musste auch der Navi3D-Client auf Linux lauffähig sein. Hierzu musste das ganze Projekt auf Linux portiert werden. Dies wurde auch genutzt, um das Projekt aufzuräumen und eine einheitliche Struktur für die einzelnen Module zu etablieren. Weiterhin wurde die Gelegenheit genutzt, einige Parameter aus dem Code in eine XML-Datei auszulagern und so den Navi3D-Client konfigurierbar zu machen.

Gimmicks sind kostenfreie Erweiterungen des Systems, welche dem Benutzer spielerischen Spaß bereiten sollen. Die hier im Ansatz realisierte Erweiterung ist ein sogenanntes Lernspiel, welches dem Fahrer die Möglichkeit gibt, sein fahrerisches und insbesondere verkehrssicheres Fahren zu überprüfen und zu schulen. Im Ansatz wurde bisher die Detektion scharfer Kurven, einschließlich der Darstellung einer Warnmeldung realisiert. In den folgenden Kapiteln wird näher auf das Spielelement und die neue Klasse Gimmick eingegangen.

Ziel ist eine möglichst realistische Beleuchtung des Pfeiles. Hierzu wurde bereits im SS09 eine Simulation zur Sonnenstandsbestimmung implementiert. Diese soll in Zukunft mittels Shadern die Szene beleuchten, sowie einen virtuellen Schattenwurf des Pfeils auf der Straße ermöglichen.

2 Hauptteil

Im folgenden werden die Komponenten des Navi3D-Clienten vorgestellt. Zum einen handelt es sich hier um neue Module (Verkehrsschilderkennung, Gimmick, Fahrspurerkennung und die Systemgruppe), welche dieses Semester implementiert wurden. Zum anderen wurden Module aus dem letzten Semester weiterentwickelt (XML-Modul, Sonnenstand,Shader und POI). Die Kapitel 2.4 und 2.7 behandeln die Portierung der entsprechenden Anwendungen nach Linux und sind somit keinem Modul im eigentlichen Sinne zugeordnet.

Tobias Braun

2.1 Verkehrsschilderkennung

Im Bereich der Verkehrsschildererkennung wurde zu Beginn des Semesters das Ziel gesetzt einige ausgewählte Verkehrszeichen als solche zu erkennen. Aufgrund des deutschen Schilderwaldes haben wir uns vorerst auf einschränkende, bzw. Verbotsschilder geeinigt, welche eine runde Form haben. Das Primärziel lag somit in der Erkennung roter, runder Verkehrsschilder. Der Hauptgrund dafür ist die Wichtigkeit der Schilder, da beispielsweise ein Durchfahrtverbotsschild wichtiger ist, als Orts- oder Hinweisschilder.

Zu Beginn der Implementierungsphase wurden Farbwerte für die Erkennung eines Rottons festgelegt. Dieser muss gewisse Farbabweichungen zulassen, da Schilder im Schatten stehen oder verblichen sein können. Werden diese Grenzen allerdings zu grob gehalten, können vermeindliche Schilder detektiert werden, die keine Schilder sind. Problematisch ist dies bei ausgeblichenen Verkehrszeichen, da diese nicht in das gewählte Raster fallen. Da wir den Regelfall betrachten, werden diese nicht detektiert. Durch eine gezielte RGB Auswahl ist es somit möglich schon im Vorfeld weiterer Berechnungen Rechenzeit zu sparen, indem die Fehlerrate minimiert wird und hauptsächlich Verkehrsschilder erkannt werden. Dafür werden von uns folgende RGB Werte genutzt:

- $Rot \geq 130$
- $Grün \leq 100$
- $Blau \leq 100$

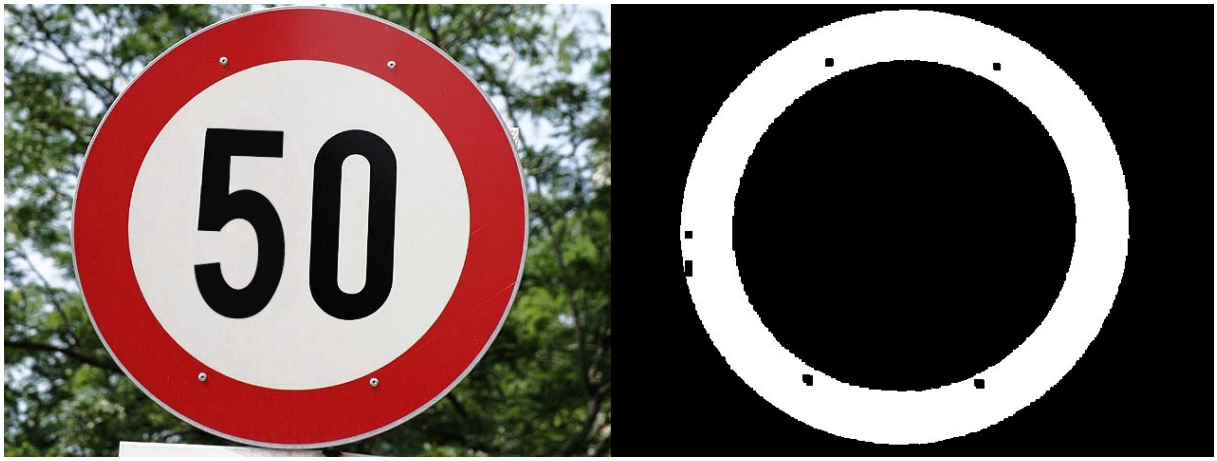


Abbildung 1: Original- und Binärbild eines Verkehrszeichens

Anhand dieser Farbwerte wird ein Binärbild erstellt. Damit aus dem entstandenen Binärbild auf ein bestimmtes Verkehrszeichen, bzw. im aktuellen Fall auf ein rundes Verkehrszeichen, geschlossen werden kann, sind noch weitere Schritte notwendig. Nicht alle Bildpunkte eines Verkehrszeichens fallen in unser gewähltes Raster. Daher wird auf das erstellte Binärbild die Dilatation angewandt, um kleinere Lücken im Bild zu füllen. Direkt im Anschluss erfolgt die Erosion, um wieder auf die Ursprungsgröße des Bildes zu kommen. In Abbildung 1 ist auf der linken Seite das Originalbild und auf der rechten Seite das nach gerade beschriebenem Verfahren erstellte Binärbild abgebildet.

Unsere eigene Implementierung wurde aufgrund der zur Verfügung stehenden Zeit verworfen. Die Implementierung umfasste bis dahin die Erosion und Dilatation, jedoch beschränkt auf Eingabebilder im RAW-Format mit 3 Farbkanälen und einem Bildformat von 640 zu 480 Bildpunkten. Daneben auch eine Teilimplementierung einer Detektion für runde Verkehrszeichen. Da sich unter Anderem die Debugmöglichkeiten aufgrund der Bildgröße stark in Grenzen hielt, wurde eine andere Möglichkeit gesucht. Die Chance auf ein umfangreiches Framework zurückzugreifen bietet die seitdem eingesetzte, kostenfreie Computer Vision Bibliothek *OpenCV*. Das Framework unterstützt, anders als unsere Implementierung, diverse Eingabeformate, was den Rechenaufwand für die Konvertierung des Eingangsbildes in das RAW-Format überflüssig macht. Die bisherige Implementierung setzten wir in Folge dessen erneut mit OpenCV um.

Anschließend werden von dem Binärbild ausgehend potentielle Kreiszeichen erkannt. Die Erkennung erfolgt mit Hilfe der in OpenCV integrierten Hough-Transformation (vgl. Abschnitt A.1). Die erkannten Kreisformen können zu Debugzwecken auf dem Originalbild ausgegeben werden. Dies ermöglicht das einfache Nachvollziehen der erkannten Verkehrszeichen. Im späteren Verlauf werden diese Ausgaben nicht zu sehen sein, da das Ziel die Projektion der Verkehrszeichen ist. Wir haben uns zum jetzigen Zeitpunkt dazu entschlossen, alle erkannten Schilder am unteren Bildrand einzufügen. Zusätzlich werden die Verkehrszeichen auf eine einheitliche Größe skaliert, auch wenn sie in Wirklichkeit unterschiedlich großund somit höchstwahrscheinlich auch unterschiedlich weit entfernt sind. Das Ergebnis der Implementierung ist in Abbildung 2 zu sehen.

Abbildung 2: Verkehrszeichenerkennung auf einer Autobahn

2.2 Gimmick

2.2.1 Idee

Um das Prinzip der 3D Navigation interessanter zu gestalten, entwickelten wir die Idee eines verkehrssicheren Spieles auf Basis der 3D-Navigation im Auto. Das Spiel soll ein Lernspiel sein, dass wie unterhaltungsorientierte Computerspiele dem Spieler Spaß bereiten sollte, jedoch das primäre Ziel dazu führt, den Autofahrer verkehrssicher durch die Straßen zu führen. Zusätzlich wird noch ein Lerneffekt erzielt, indem der Fahrer mit den Verkehrsregeln konfrontiert wird und diese einhalten muss um Punkte zu sammeln.

Der Spieler hat zwar große Freiheiten, den Ablauf des Spieles selbst zu bestimmen, Kontrolle übt der Spieler in der Wettbewerbsituatuion aus (besser zu sein als Andere Spieler oder als die eigene letzte Fahrt), die durch globale und lokale Leaderboards erzielt werden kann. Durch ein Leaderboard wird die Motivation gestärkt und die Lerninhalte auf eine getarnte Art und Weise vermittelt.

Bei dem Spielkonzept werden Münzen auf die Fahrbahn gelegt, die den richtigen Weg weisen (siehe Abbildung 3). Kommt das Auto der Münze entgegen wird sie eingesammelt und dem Punktestand hinzugefügt. Extrapunkte kann man sammeln, indem man den Sicherheitsabstand einhält und regelmässige Pausen, bei langen Fahrten macht. Der Anreiz Punkte für sicheres Fahren zu sammeln, könnten möglicherweise Verkehrrsünder dazu bewegen, sich an die Verkehrsregeln zu halten. Um den Ehrgeiz weiter zu erhöhen kann noch ein lokales und ein globales Leaderboard am Ende der Fahrt angezeigt werden, damit die Fahrer ihre eigenen Highscores verbessern und ihr fahrerisches Können mit anderen messen können. Ein weiterer Einsatzort für das Spielkonzept wäre bei Fahrschulen. Fahranfänger könnten spielerisch und mit Spaß das Autofahren erlernen.

Abbildung 3: Lernspiel 3D-Navigation

2.2.2 Konzept

Durch fahrerisches Können und degressives Fahren soll der Fahrer mit Punkten belohnt werden. Verkehrswidrigkeiten werden hingegen mit Punktabzügen bestraft. Ziel des Spiels ist es bis zur Ankunft am Zielort die maximal mögliche Anzahl an Punkten zu erreichen. Eine Einnahmequelle von Punkten stellen so genannte Münzen dar. Vor Fahrtbeginn werden die Münzen in gleichmässigem Abstand zueinander auf der zu fahrenden Strecke verteilt. Der Fahrer sammelt auf seiner Route liegende Münzen durch Überfahren mit dem Auto ein. Eine Münze repräsentiert einen Punkt. Die Anzahl gesammelter Münzen entspricht dem Basispunktewert. Dieser kann durch Boni nochmals vervielfacht werden. Ein Bonus stellt demnach einen Multiplikationsfaktor des Basispunktestands dar. Durch vorausschauendes und umweltfreundliches Fahren soll dem Fahrer die Möglichkeit gegeben werden seinen Punktestand um ein vielfaches zu erhöhen. Zu den Gegebenheiten, welche jeweils einen Bonus verleihen, zählt unter anderem das Einhalten von Pausen bei längeren Fahrten. Hierbei ist vorstellbar, dass dem Fahrer bspw. nach der Hälfte der gefahrenen Strecke vom Navigationsgerät ein Rastplatz vorgeschlagen wird, zu dem er sich optional navigieren lassen kann. Nimmt der Fahrer den Vorschlag an und stellt auf dem Rastplatz sein Fahrzeug für eine vom Navigationsgerät berechnete, angemessene Zeit ab, erhält er einen Bonus. Eine weitere Möglichkeit einen Bonus zu erhalten bietet das Ausschalten des Motors bei längeren Standzeiten. Insbesondere vor Bahnübergängen, aber auch bei Staus wird der Fahrer durch einen Bonus für sein umweltfreundliches Verhalten belohnt. Nach dem Zuckerbrot und Peitsche-Prinzip soll der Fahrer nicht ausschließlich für gutes Fahren belohnt werden, sondern aufgrund von verkehrswidrigem Verhalten im Straßenverkehr auch Bestrafungen in Form von Punktabzügen erhalten. Vernachlässigt der Fahrer den Sicherheitsabstand zum Vorausfahrenden, missachtet er Stop-Schilder, indem er nicht an der Haltelinie zum Stehen kommt oder fährt er in verkehrsberuhigten Bereichen nicht die vorgeschriebene Schrittgeschwindigkeit drohen ihm große Punktverluste.

Das Einhalten der genannten Verkehrsregeln ließe sich über eine Videoschnittstelle (Schildererkennung bzw. Sicherheitsabstand) und die über Satellit gegebene Möglichkeit der Geschwindigkeitsmessung überprüfen. Mit gleichen Hilfsmitteln ließe sich auch die aktuell gefahrene und die tatsächlich vorgeschriebene Geschwindigkeit ermitteln und miteinander vergleich, wobei Geschwindigkeitsübertretungen ebenfalls mit Punktabzügen bestraft werden sollten. Nachdem der Fahrer sein Ziel erreicht hat wird der fnale Punktestand einschließlich Boni und Punktabzügen berechnet. Da die Länge der gefahrenen Routen oftmals variiert und um den Gesamtpunktestand einer Fahrt mit vorherigen Highscores im Leaderboard miteinander vergleichen zu können, wird ein relativer Gesamtpunktestand berechnet. Zur Berechnung dieses Endpunktestands ist die gefahrene Strecke mit den für die Fahrt erhaltenen Punkten in Bezug zu setzen. Anschließend kann der prozentuale erhaltene Wert entweder direkt als Zahlenwert mit anderen Scores verglichen werden oder aber in bildlicher Form. Beispielsweise könnte der prozentuale Punktestand in Form von Sternsymbolen auf einer Skala von 1 bis 10 gemessen werden, wobei 10 Sterne die bestmögliche fahrerische Leistung repräsentieren würden.

Pausen bei längeren Fahrten sollen vom Navigationssystem eigenständig berechnet und dem Fahrer in Form von Raststätten vorgeschlagen werden. Steuert der Fahrer eine solche Raststätte an und parkt dort für einen festgelegten Zeitraum sollen ihm Extrapunkte gutgeschrieben werden. Ebenfalls neu ist die Idee das Einhalten des Sicherheitsabstandes zum Vorrausfahrenden zu Belohnen, das Unterschreiten hingegen durch Punktabzug zu bestrafen. Die Belohnung soll in regelmäßigen Zeitintervallen durch Punktgutschrift erfolgen, sofern der Fahrer in besagtem Zeitintervall stets den Mindestsicherheitsabstand eingehalten hat. Zudem soll der Fahrer belohnt werden, wenn er sich kontinuierlich an das Rechtsfahrgebot hält. Mit einer Kamera soll ermittelt werden, ob sich vor dem Fahrer weitere Fahrzeuge auf Fahrspuren rechts von ihm

befinden. Ist dies nicht der Fall und fährt der Fahrer dennoch nicht auf der möglichst rechten Spur werden ihm im Abstand von fünf Minuten Punkte abgezogen. Hält der Fahrer sich hingegen an das Rechtsfahrgebot werden ihm im Intervall von 15min Punkte gutgeschrieben.

Es gibt mehrere graphische Elemente, welche in dem Spiel unterschiedliche Aufgaben erfüllen. Die bereits zuvor erwähnten Münzen symbolisieren die Punkte aus denen sich der Basispunktestand am Ende der Fahrt zusammensetzt. Des weiteren soll das Ziel der Strecke durch einen dreidimensionalen vom Himmel senkrecht herabzeigenden Pfeil markiert werden. Da die zu fahrende Strecke durch die auf ihr liegenden Münzen womöglich nicht ausreichend repräsentiert wird, um dem Fahrer anzuzeigen, wohin er genau fahren muss, soll die Route direkt auf der Straße farblich mit einer zweidimensionalen Fläche bzw. Fahrbahn markiert werden. Dies ermöglicht dem Fahrer auch beim Abweichen von der ursprünglich berechneten Münz-Strecke sein Ziel zu erreichen. Eine weitere Idee besteht darin vor dem realen Auto ein kleines virtuelles Auto einzublenden, welches vor dem realen Auto auf der Straße fährt und mit dessen Hilfe die Münzen eingesammelt werden können.

Achim Brauer, Johannes Seidel

2.3 Fahrspurerkennung

Um die Navigation weiter zu verbessern, war für dieses Semester geplant, eine Fahrspurerkennung zu implementieren. Hierdurch soll der Nutzer noch genauer zum Ziel navigiert werden, indem er zusätzlich zu den bisherigen Informationen Auskunft über seine aktuelle und die benötigte Fahrspur erhält.

Ziel für dieses Semester war, mit Hilfe einer im Auto montierten Kamera Fahrspuren zu erkennen und zu klassifizieren. Um die generierten Daten weiter zu verwenden, sollte diese Fahrspurerkennung außerdem in das bestehende System eingebunden werden.

Für die Bildverarbeitung und Bildbearbeitung wurde die Bibliothek „Open Source Computer Vision“ (kurz: OpenCV) verwendet. Diese muss vor der Verwendung in dem Projekt installiert und mit dem Projekt verknüpft werden. Die Bibliothek stellt unter Anderem eine Reihe von Funktionen bereit, die wir zum Extrahieren der wichtigen Informationen benötigten. Zur Extraktion dieser Informationen aus dem Video wird jedes Bild einzeln untersucht. Hierfür wird für jedes Bild eine Region of Interest (kurz: ROI) bestimmt, welche angibt, in welchem Bereich des Bildes nach dem Fahrspurmarkierungen gesucht werden muss. Da uns besonders der mittlere Teil des Bildes interessiert, hat die ROI nur 45% der Gesamthöhe und 100% der Gesamtbreite. Diese Werte sind nötig da in dem unteren Teil des Bildes Teile vom Auto und im oberen Teil des Bildes nur Himmel zu sehen ist. Wenn die ROI nicht bestimmt wird, können in diesen Bereichen falsche Ergebnisse auftreten. Das nun verkleinerte Bild wird danach in ein Grauwertbild umgewandelt und mit Hilfe des Gauß-Filters geglättet. Bevor nun die Erkennung der Markierung durchgeführt wird, werden die Kanten des Bildes durch den Canny-Edge-Detector hervorgehoben. Die Erkennung der Markierung erfolgt mit der Funktion „HoughLines“. Diese Funktion filtert die Linien mit Hilfe von Hough-Transformationen. Die Linien werden dann in der Hess'schen Normalform abgespeichert.

$$r = x \times \cos\vartheta + y \times \sin\vartheta$$

Wobei r der Abstand der Geraden zum Ursprung und ϑ der Winkel zwischen der x-Achse und der Senkrechten der Gerade darstellt.

Damit weder die völlig waagrechten noch die senkrechten Linien als Fahrspurmarkierung erkannt wird (es gibt keine waagrechten Markierungen und durch die perspektivische Verzerrung des Bildes auch keine Senkrechten), müssen diese heraus gefiltert werden. Hierfür wurde eine Beschränkung eingebaut, dass

$5° \leq \vartheta \leq 75°$ oder $115° \leq \vartheta \leq 175°$ (siehe Abbildung 4)

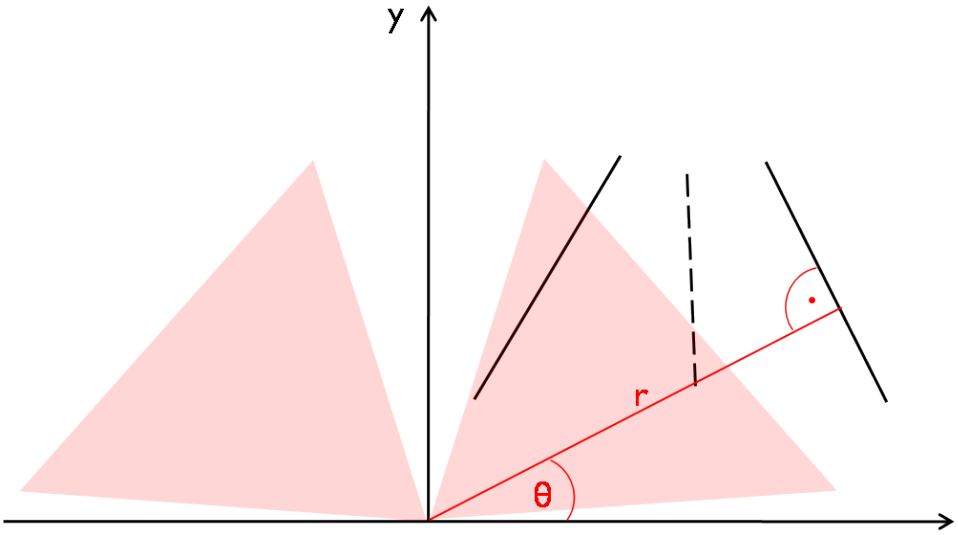


Abbildung 4: Beschränkung von Theta

Die gefundenen Linien werden daraufhin in eine Liste gespeichert. Wobei hier darauf geachtet werden muss, dass nicht mehrere Linien für nur eine Fahrspurmarkierung gefunden werden. Um dies zu verhindern wurde eine Klassifizierung eingeführt. Diese Klassifizierung sorgt dafür, dass Linien, die sich sehr ähnlich sind ($\vartheta \leq 19°$ oder $r \leq imagewidth/8$) nur einmal in der Liste gespeichert werden. In jedem Frame wir dann für die gefundenen Linien überprüft, ob diese bereits in der Liste vorhanden sind. Wenn dies der Fall ist, wird die Linie in der Liste aktualisiert, wenn nicht, wird sie in die Liste eingefügt. Zusätzlich hierzu hat jede Linie einen Zähler, der zählt, wie lange die Linie schon nicht mehr gefunden wurde. Wenn diese Zähler einen Schwellwert überschreitet, wird die Linie aus der Liste gelöscht. Dies wird gebraucht, damit kein „Flackern“ im Bild entsteht, wenn eine Linie in einem Frame nicht gefunden wird.

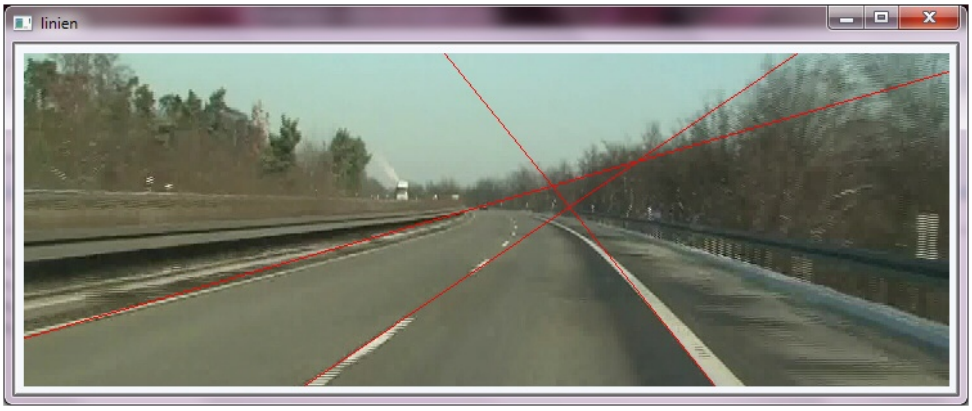


Abbildung 5: Ergebnis

Im letzten Schritt wurde die Fahrspurerkennung in das bestehende WINDOWS-System eingebunden. Die Lib- und include-Files für OpenCV wurden in Perforce eingepflegt. In die Datei „define.h“ wurde eine Flag `FAHR` eingebaut, wodurch sich die Fahrspurerkennung ein- und ausschalten lässt. Wird das Flag aktiviert, muss das Video der Testfahrt (unter „Framework\Navi_NG3_Demofiles\03_Autobahn_Schoefferstr.avi“) auf dem ausführenden PC vorhanden sein. Neben der normalen Pfeildarstellung öffnet sich dann ein

zusätzliches Fenster, welches das Video mit den eingeblendeten Fahrspuren zeigt.

Die Klasse `Navi3D_Fahrspur` besitzt eine Liste mit den aktuellen Fahrspurmarkierungen.

Listing 1: Listenstruktur

```
list<lane> lanes;

struct lane{
        float rho;
        float theta;
        CvPoint pt1;
        CvPoint pt2;
        int id;
        int counter;
        bool erased;
};
```

Die Variablen `rho` und `theta` sind hier die schon zuvor erwähnten Variablen aus der Hess'schen Normalform. `pt1` und `pt2` sind zwei Punkte, durch die die Gerade durchgeht. Die `id` ist die eindeutige ID der Fahrspurmarkierung, der `counter` gibt an, seit wie vielen Frames die Fahrspurmarkierung nicht mehr gefunden wurde und das Flag `erased` wird auf true gesetzt, wenn die Fahrspurmarkierung, nachdem sie lange nicht mehr erschienen ist, gelöscht wird.

Für die Fahrspurerkennung gibt es noch mehrere Erweiterungs- und Verbesserungsmöglichkeiten. Grundsätzlich gibt es drei Arten von Fahrspurmarkierungen: Durchgezogene Linie, gestrichelte Linien oder eine Kombination aus beiden. Die Farbe der Markierung richtet sich nach der Umgebung. So ist üblicherweise die Markierung weiß, bei Baustellen beispielsweise ist sie jedoch gelb und als vorrangig zu betrachten. Diese Unterschiede müssten in einem nächsten Schritt noch in die bestehende Implementierung eingebaut werden. Außerdem können bei der Erkennung der Fahrspur einige umgebungsbedingte Störungen entstehen:

- Schatten
- Feuchtigkeit (evtl. Spiegelungen)
- Fehlende Markierungen

Diese Faktoren sollten im weiteren Verlauf der Lösung der Aufgabe möglichst berücksichtigt werden, wobei dabei beispielsweise auf die Vorberechnung des zukünftigen Verlaufs der Markierung zurückgegriffen werden kann.

Julia Runge, Antje Gloystein

2.4 Portierung Grafik

Im folgenden Kapitel wird die Portierung des Navi3D-Clienten auf Linux beschrieben. Die eigentliche Arbeit des Portierens liegt hier nicht im Mittelpunkt. Es wird viel mehr erläutert, wie das Projekt unter Linux einzurichten ist. Der Code selbst kann im Anhang (A.4) eingesehen werden. Wichtige änderungen wurden dort mit einem Kommentar versehen.

2.4.1 Linux und Eclipse einrichten

Für die Portierung des Navi3D Clienten wurde Open Suse 11.1 (32 Bit) verwendet. Grundlegend sollte jede akutelle Linux distribution verwendet werden können. Die hier beschriebene Vorgehensweise richtet sich nach Open Suse 11.1. Bei einer anderen Distribution könnten gewisse Dinge anders geregelt oder

aufgebaut sein. An dieser Stelle soll auch darauf hin gewiesen werden, dass VirtualBox nicht für die Entwicklung verwendet werden kann. Hier gibt es Probleme mit der 3D Beschleunigung. Möglichwerweise funktioniert dies besser mit VMWare, dies wurde aber nicht mehr getestet, da das Linuxsystem für die Portierung letztendlich auf einer externen Festplatte installiert wurde. So konnten etwaige Probleme induziert durch eine Virtualisierungslösung ausgeschlossen werden. Als Entwicklungsumgebung wird Eclipse mit der entsprechenden Erweiterung für C++ verwendet. Unter http://www.eclipse.org/downloads/ kann die entsprechende Version für Linux heruntergeladen werden (eclipse-cpp-galileo-SR1-linux-gtk.tar.gz).

2.4.2 Bibliotheken

Der Navi3D Client benötigt verschiedene Bibliotheken. Normalerweise können diese einfach mit Yast2 oder zypper (Komandozeile) installiert werden. Die folgende Tabelle listet die Verschiedenen Bibliotheken auf:

Bibliothek	Package Name
glut	freeglut, freeglut-devel
cv	opencv, opencv-devel
highgui	siehe cv
mysqlclient	libmysqlclient15, libmysqlclient_r15, libmysqlclient-devel
SDL_image	SDL_image, SDL_image-devel
SDL	SDL, SDL-devel
GL	Standardmäßig installiert
pthread	Standardmäßig installiert

Tabelle 1: Bibliotheken für den Navi3D-Clienten unter Linux

Wenn alle Bibliotheken installiert sind, kann Eclipse einfach auf sie zugreifen. Es müssen nur die ensprechenden Libs Eclipse bekannt gemacht werden. Hierfür geht man unter Window auf Preferences und navigiert im folgenden Fenster zu C/C++ und Settings. Hier wählt man die GCC C++ Linker-Einstellungen aus und trägt die ganzen Bibliotheken aus der Tabelle 1 wie im Bild 6 ein.

2.4.3 Konfiguration

Um auf die Projektdateien zugreifen zu können muss das entsprechende Perforce-Plugin installiert sein. Das Perforceplugin wird einfach über die Pluginverwaltung von Eclipse nachinstalliert. Das Perforceplugin muss mit den entsprechenden Daten konfiguriert werden. Die IP-Adresse ist von einem Laboringeneur in Erfahrung zu bringen. Mit dem entsprechenden Benutzerdaten kann man dann das Projekt (Navi3D_Client) herunterladen. Hierbei sollte darauf geachtet werden, den Pfad ensprechend dem Workspacepfad von Eclipse zu wählen. Falls das Projekt in Eclipse nicht sichtbar wird, muss man dies erst im Perforceplugin in den Workspace importieren. Leider sichert Eclipse nicht das Arbeitsverzeichnis in den Projekteinstellungen, es ist also nötig in den Einstellungen das Arbeitsverzeichnis auf ./RUN zu stellen. Hierfür wird in den Run Configurations für Debug und Release entsprechend dem Bild 7 konfiguriert. Es kann auch vorkommen, dass unter einer anderen Linuxdistribution die Codierung der Dateien anders eingestellt wurde. Dies macht sich durch Quellcodedateien bemerkbar, bei denen jede Zeile als fehlerhaft angezeigt wird. Um dieses Problem zu beheben muss in Eclipse UTF-8 als Zeichencodierung eingestellt werden. In schwierigen Fällen muss der Inhalt der Datei in eine neue Datei kopiert werden. Aus welchem Grund dies manchmal nötig ist, konnte nicht genau festgestellt werden.

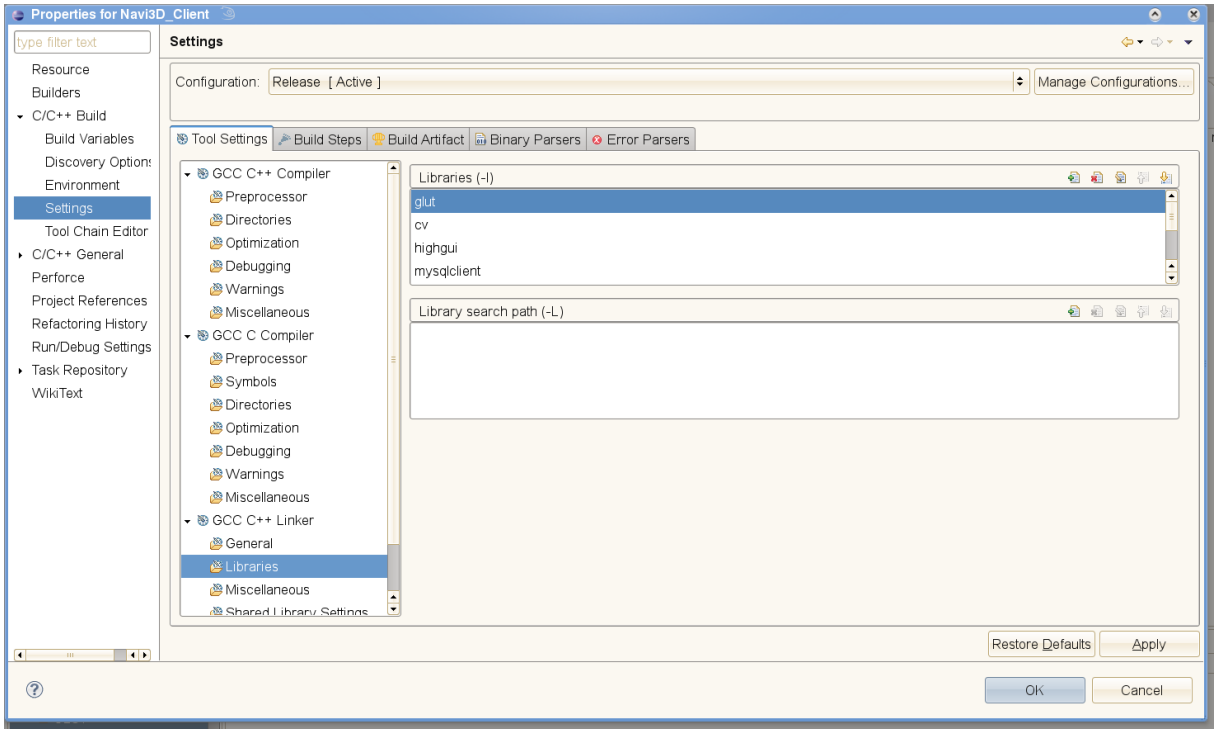


Abbildung 6: Bibliotheken in Eclipse einbinden

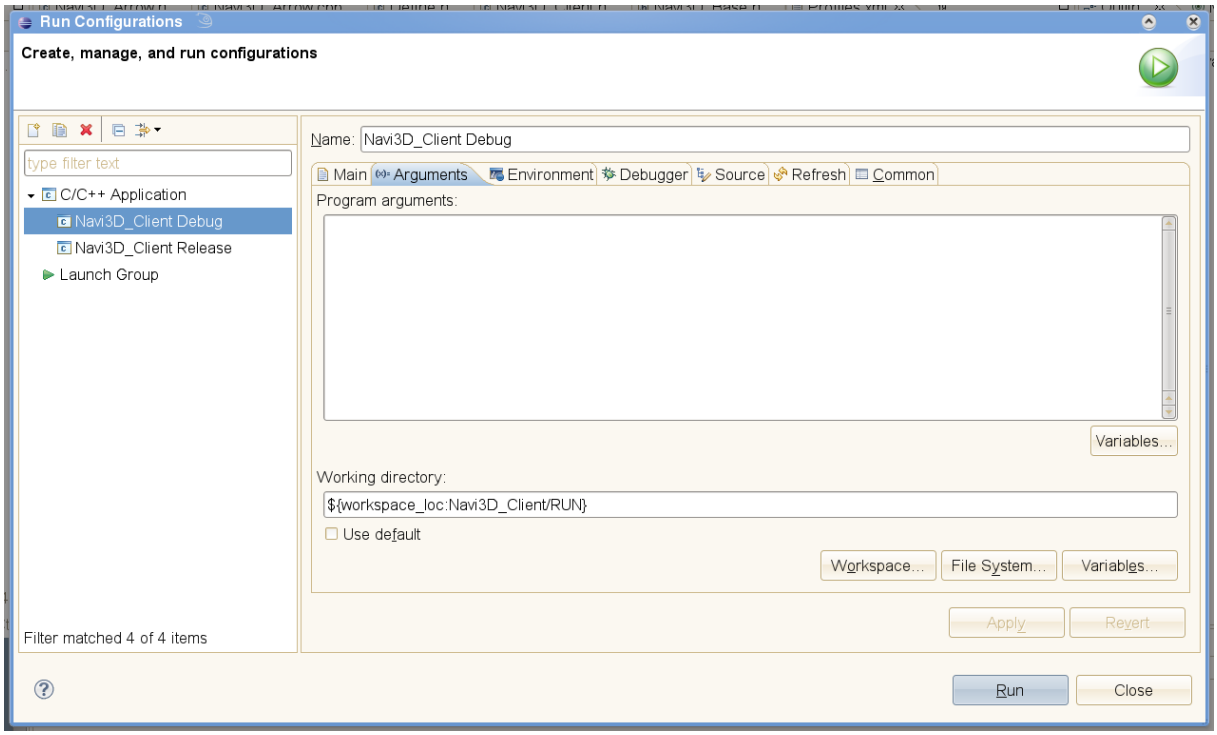


Abbildung 7: Arbeitsverzeichnis einstellen

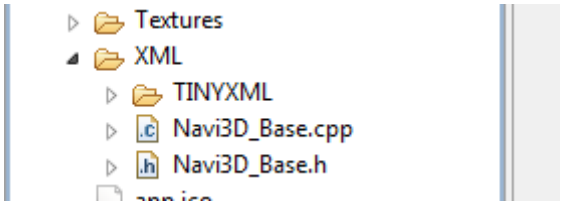


Abbildung 8: Neue Projektstruktur

2.4.4 Projektstruktur

Die Portierung wurde auch dazu genutzt, das Projekt aufzuräumen und eine neue Struktur zu etablieren. Jedes Modul hat seinen eigenen Unterodner im Projektordner (z.B. Navi3D-Client/XML/..). In diesem Ordner liegen alle Sourcecodedateien (Header und cpp-Dateien) zu diesem Modul. Nötige Dateien, welche zur Laufzeit benötigt werden, liegen im Ordner RUN (Profiles.xml, etc.).

Tobias Braun

2.5 XML-Modul

Gegenüber dem letzten Semester wurden einige Erweiterungen am XML-Modul vorgenommen. Diese werden nun vorgestellt. Da der Entwickler nach diesem Semester mit dem Masterprojekt fertig ist, wird in diesem Kapitel auch auf die Erweiterung des XML-Moduls eingegangen. Dies soll sicher stellen, dass es weiteren Teilnehmern des Masterprojekts einfach möglich ist, das XML-Modul für die eigenen Zwecke zu erweitern.

2.5.1 Neue Parameter in der XML Datei

Neben dem im letzten Semester eingeführten Abschnitten “Bases“ und „Profiles“ mit den verschiedenen Basispunkten und Profilen für die Pfeildarstellung ist nun noch der Abschnitt „System“ mit verschiedenen Unterabschnitten und Parametereinstellungen für den Navi3D-Clienten dazu gekommen. Unter System sind die folgenden Unterabschnitte zu finden:

SQL_DB - Parameter für die Konfiguration der SQL-Datenbank:

- IP und Port der Datenbank
- User und Password und der Name der Datenbank (db_name). Da die SQL-Schnittstelle noch in Entwicklung ist, können an dieser Stelle nicht mehr Informationen dazu gegeben werden.

GADA - Parameter um die Art der Pfeildarstellung zu Konfigurieren:

- IP und Port (ist standardmäßig 7000) um das Target oder die Beckerdemo anzusprechen. Mode schaltet zwischen den verschiedenen Modi durch:
  - 0 - STANDARD_ARROW - Ein statischer Pfeil welcher für einfache Präsentationen und Debugging verwendet werden kann.
  - 1 - BECKER_DEMO - aktiviert die Kommunikation mit der Beckerdemo. Da die Beckerdemo nur unter Windows lauffähig ist, muss IP und Port entsprechend konfiguriert werden, um eine Verbindung mit dem Windowsrechner herstellen zu können.

  - 2 - TARGET - aktiviert die Kommunikation mit dem alten Target. In der aktuellen Version kann diese Option nicht genutzt werden, da die Kommunikation mit dem Target neu implementiert werden muss (Message Parsing). Der Parameter wurde aber schon implementiert, um die spätere Erweiterung an dieser Stelle in die Konfiguration einbinden zu können.

- Die Option WAIT_FOR_CONNECT lässt sich über den Parameter wait ein (1) und ausschallten (0).

KERNEL - Parameter um Systemnahe Einstellungen vorzunehmen:

- Mit "loglevel" kann die Menge der Informationen, welche in das Logfile geschrieben werden, konfiguriert werden:
  - 3 schreibt alle Informationen raus (Einfache Informationen und Debuginformationen).
  - 2 ist für mittelwichtige Informationen.
  - 1 steht für einfache Informationen.
  - 0 Schaltet das Loggen aus.

- FOG - Verschiedene Parameter um den Nebel bei der Pfeildarstellung zu konfigurieren. In der aktuellen Version funktioniert der Nebel nicht mit den Shadern. Der Nebel kann also nur genutzt werden, wenn die Shader über das entsprechende Define in der Define.h ausgeschaltet werden.
  - ON - Schaltet den Nebel an (1) oder aus (0)
  - FOG_START - in welcher Entfernung von der Kamera der Nebel beginnen soll (Standard: 50).
  - FOG_END - in welcher Entfernung von der Kamera der Nebel enden soll (Standard: 100).

- INTERPOLATE_POINTS - Verschiedene Parameter um die Interpolation zwischen den Stützpunkten des Pfeils zu konfigurieren.
  - ON - Schaltet die Interpolation an (1) oder aus (0)
  - INTERPOLATION_RESOLUTION - konfiguriert die Auflösung, mit der interpoliert wird (Standard: 50)

- COLOR_ON_ARROW_SPEEDCONTROL - Parameter um die Einfärbung des Pfeils bei scharfen Kurven zu konfigurieren.
  - ON - Schaltet die Pfeileinfärbung an (1) oder aus (0)
  - TEST_SPEED - wird genutzt um im STANDARD_ARROW oder BECKER_DEMO Modus bei scharfen Kurven die Färbung des Pfeils sehen zu können. Da in diesen Modi keine Geschwindigkeitsinformationen vorliegen, kann diese über diesen Parameter fest vorgegeben werden (Standard: 100 in km/h).

- STEREOPROJECTION - schaltet die Stereoprojektion an (1) oder aus (0)

Die Vorgestellten Parameter wurden aus der Define.h ausgelagert um eine einfachere Konfiguration des Navi3D-Clienten zu gewährleisten. Bisher musste für eine Änderung immer erst das gesamte Projekt neu übersetzt werden. Nun lädt die ausführbare Datei die Parameter beim Start und konfiguriert sich entsprechen. Die gesamte XML-Datei kann in (A.5.3) gefunden werden.

2.5.2 Erweiterung des XML-Moduls um eigene Parameter

Das folgende Beispiel soll veranschaulichen, wie das XML-Modul erweitert werden kann. Der Code des XML-Moduls kann in A.5 eingesehen werden. Grundsätzlich lädt Tiny-XML erst die gesamte XML-Datei in den Speicher. Über die Tags können nun die verschiedenen Parameter abgefragt werden. Diese werden in einem systemweit verfügbaren Struct (N3D_System) zur Verfügung gestellt. Diese Structs sind in der „Navi3D_Base.h“ zu finden und können entsprechend erweitert werden (N3D_KERNEL, N3D_GADA und N3D_SQL_DB). Über die Funktion „N3D_System getSystemParameters(void)“ kann das Struct von der Klasse „Navi3D_Base“ abgefragt werden. Wie die Profile und Basispunkte eingelesen werden, wird hier nicht erläutert, da dies schon in [BBB+09] abgehandelt wurde. Über die Funktion „void Navi3D_Base::loadXML(string filename)“ wird die XML-Datei geladen. Das Auswerten der verschiedenen Abschnitte (Bases, Profiles und System) wird in eigenen Funktionen erledigt. Der Systemabschnitt wird mit der Funktion „void loadSystem(TiXmlHandle hDoc)“ geladen. Falls nun ein weiterer Abschnitt auf der Höhe dieser drei eingefügt werden sollte, muss hier entsprechend eine weitere Funktion eingefügt werden. Diese ist entsprechend den anderen Funktionen aufzubauen. Dazu nun mehr. Die loadSystem Funktion ist (gekürzt) wie folgt aufgebaut:

Listing 2: XML-Modul: loadSystem

```
1  //load profiles
2
3  void Navi3D_Base::loadSystem(TiXmlHandle hDoc)
4  {
5          string name;
6          TiXmlElement* pElem;
7          TiXmlElement* pSystem;
8
9          pElem=hDoc.FirstChild("Data").FirstChild("System").ToElement();
10
11         for(pElem; pElem; pElem=pElem->NextSiblingElement() )
12         {
13                 //block: SQL_DB
14                 pSystem = pElem->FirstChildElement("SQL_DB")->FirstChildElement("IP");
15                 if(pSystem)
16                 {
17                         //IP
18                         mSystem.SQL_DB.IP = pSystem->GetText();
19                         //port
20                         mSystem.SQL_DB.port = atoi(pSystem->NextSiblingElement("port")->GetText());
21                         //user
22                         mSystem.SQL_DB.user = pSystem->NextSiblingElement("user")->GetText();
23                         //password
24                         mSystem.SQL_DB.password = pSystem->NextSiblingElement("password")->GetText();
25                         //db name
26                         mSystem.SQL_DB.dbName = pSystem->NextSiblingElement("db_name")->GetText();
27                         printLog("Navi3D_Base.cpp: SQL_DB settings found",3);
28                 }
29                 else
30                 {
31                         printLog("Navi3D_Base.cpp: Error in XML-File - System SQL_DB loading",1);
32                         return;
33                 }
34 ...
35         }
36 }
```

Die gesamte Funktion kann in Anhang A.5 eingesehen werden. Zunächst werden die nötigen Variablen initialisiert, dann wird in die Variable pElem der Systemabschnitt geladen:

```
1  pElem=hDoc.FirstChild("Data").FirstChild("System").ToElement();
```

Die For-Schleife geht nun alle Elemente durch. Im Block SQL_DB wird mit

```
1 pSystem = pElem->FirstChildElement("SQL_DB")->FirstChildElement("IP");
```

Das erste Element geladen. In diesem Fall ist dies IP. Innerhalb des Blocks wird der Inhalt von IP dem System Struct zugewiesen. Dann wird mit

```
1 atoi(pSystem->NextSiblingElement("port")->GetText());
```

der Port geladen und zugewiesen. Mit diesem Schema werden alle Parameter geladen. So können weitere Parameter dem XML-Modul hinzugefügt werden. Weitere Informationen können in [Tho10] nachgelesen werden.

Tobias Braun

2.6 Sonnenstand und Shader

2.6.1 Sonnenstand (Rueckblick)

Um eine realistische, dem aktuellen Sonnenstand nachempfundene Beleuchtung zu realisieren, wurde im SS09 die Klasse „sunpos“ implementiert. Zur Bestimmung des Azimuth- und des Elevation- Winkels, werden innerhalb dieser Klasse verschiedene vereinfachte Berechungen durchgeführt(siehe Abbildung 9). Die resultierenden Winkel müssen zum Schluss noch in Kartesische Koordinaten überführt werden, damit wir sie zur Positionierung einer Lichtquelle im 3D Raster verwenden können.(Siehe Abschlussbericht SS09)

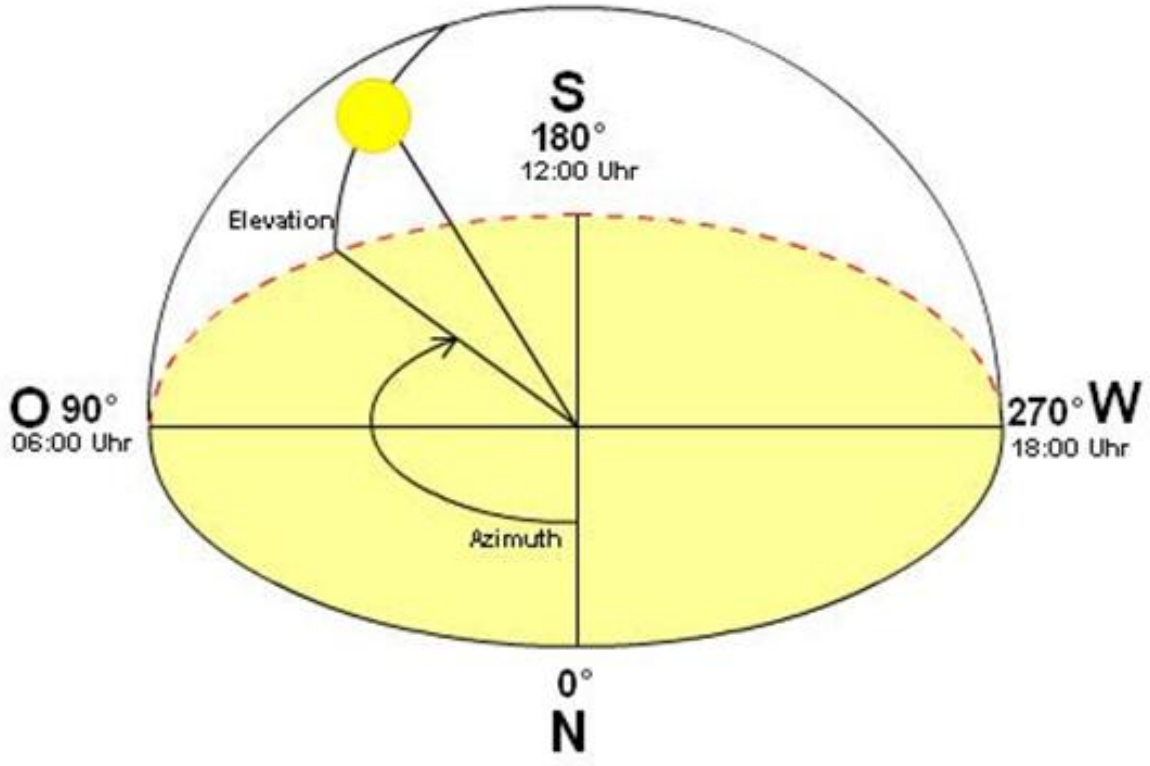


Abbildung 9: Sonnenstand

2.6.2 Heading

Der Sonnenstand wird statisch in seiner Bahn, von Osten nach Westen berechnet. Fährt das Fahrzeug nach Norden entspricht dies einem Verlauf von rechts(Osten), über hinten(Süden), nach links(Westen). Jedoch fährt das Fahrzeug nicht immer nach Norden. Da in der Navi3D Simulation, entsprechend der Fahrtrichtung, der Pfeil gedreht wird und nicht das Fahrzeug(Oder Koordinatensystem), bleibt bei einer Kurve die Sonne statisch am gleichen Platz. Hierzu gibt es einen Heading-Winkel, welcher aus den GPS Daten an übermittelt wird. Dieser gibt den Winkel zwischen Fahrtrichtung und Norden an. Mit Hilfe dieses Winkels wird die Sonne an die benötigte Position (Rotation um Y-Achse) rotiert. Leider funktioniert dies bisher nur als Demo, da in den Aufgezeichneten Daten, sowie in der Becker Demo, dieser Winkel nicht mitgeliefert wird. Im realen Target ist er jedoch vorgesehen und sollte somit auch funktionieren.

2.6.3 Shader

Laden individueller Shader:
Innerhalb der Klasse „shader“ können neue Pixel- und Vertex- Shader in Form von Textdateien durch die Funktion LoadShader() eingebunden werden.

Bsp:

Listing 3: Shader Laden

```
1        GLhandleARB VS_Pfeil; //vertex shader
2        GLhandleARB PS_Pfeil; //pixel shader
3        LoadShader(&shader_Pfeil, &VS_Pfeil, &PS_Pfeil, "Shader//VS_Pfeil.txt","Shader//PS_Pfeil.txt")
             ;
```

Aktivierung:
Die alte OpenGL-Beleuchtung kann mittels DEFINE durch eine Shader-Beleuchtung erweitert werden(siehe define.h).

Debuggen von Shadern:
Um die eingebundenen Shader debuggen zu können, müssen Haltepunkte in der Datei shader.cpp gesetzt werden(siehe Abbildung 10). Dies ist nötig da die Kompilierung der Dateien zur Laufzeit geschieht und die eventuellen Fehlermeldungen im Konsolenfenster nur schwer zu finden sind. Durch setzen der Haltepunkte bleibt die Ausführung direkt bei der Fehlerausgabe stehen. Um auf alle möglichen Fehlervarianten zu reagieren, müssen vier Haltepunkte innerhalb der vier „if(!success)“ Bereiche gesetzt werden. Idealerweise jeweils nach der Zeile „cout¡¡infoLog¡¡endl;“

Bsp:
Ein Semikolon Fehler im Vertex-Shader:

Listing 4: Auszug aus Vertex Shader

```
1  Gl_Position = ftransform() //<-Hier fehlt ein Semikolon
2  gl_FrontColor = gl_Color;
```

Führt zu folgender Fehlermeldung im Navi3d Konsolenfenster:

Implementierte Shader für Navigationspfeil:

Zur Zeit stehen zwei leicht unterschiedliche Shader für den Navigationspfeil zur Verfügung. Der erste entspricht einer normalen phong-Beleuchtung, ist aber zurzeit nicht aktiv(siehe Abbildung 12).

Dateien: PS_phong.txt und VS_phong.txt (Navi3D_Client\Shader)

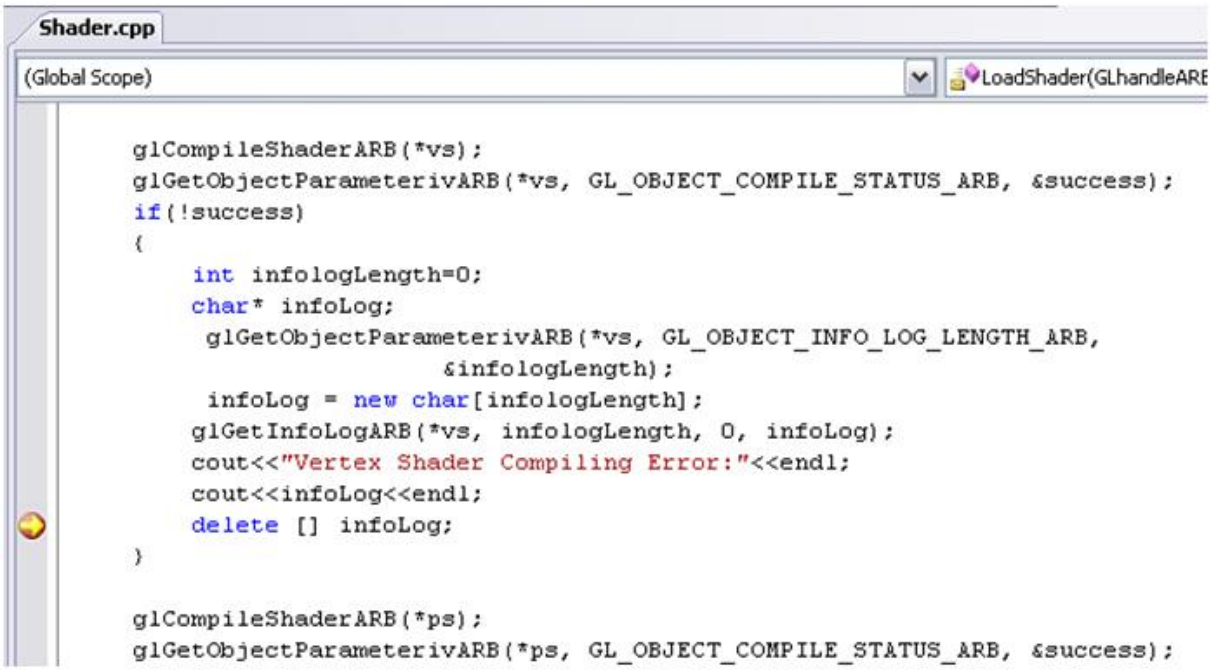


Abbildung 10: Breakpointplatzierung

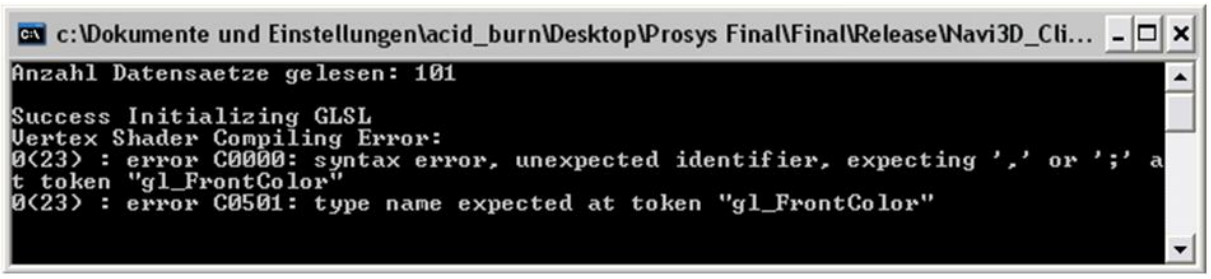


Abbildung 11: Debugausgabe im Konsolenfenster

Der zweite zur Zeit im Projekt aktivierte Shader entspricht einer Abwandlung von Phong(siehe Abbildung 13).

Dateien: PS_Pfeil.txt und VS_Pfeil.txt (Navi3D_Client\Shader)

Schattenwurf:

Mittels einer Shadowmap wird der jeweilige Schattenwurf zu jedem Sonnenstand berechnet. Damit dieser auch dargestellt werden kann, wurde ein schwarzer (somit unsichtbarer) Boden unter dem Pfeil platziert. Der berechnete Schatten kann somit in einem Grauton (nicht schwarz) darauf projiziert werden(siehe Abbildung 14).

2.6.4 Sonstiges

Nachtmodus:

Um bei Nacht keine störenden Lichteffekte zu erhalten wird bei einem Sonnenstand von unter 20 Grad oberhalb des Untergrundes die spiegelnde Reflexion aus geschaltet.

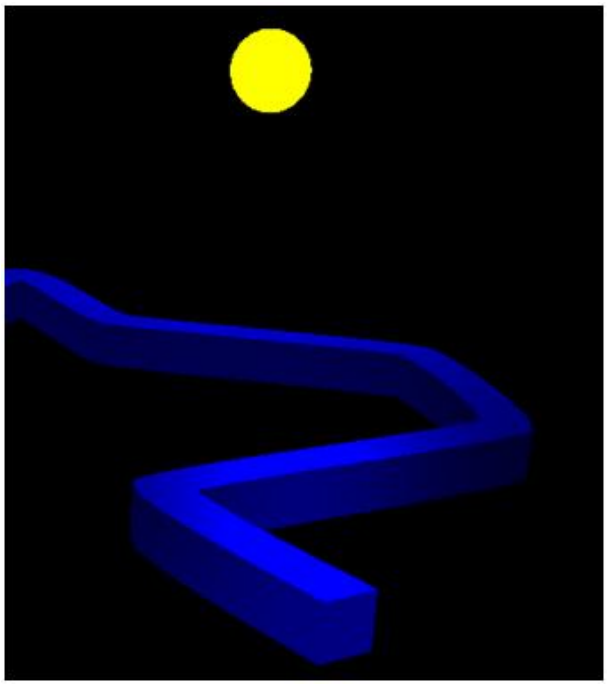


Abbildung 12: Phong

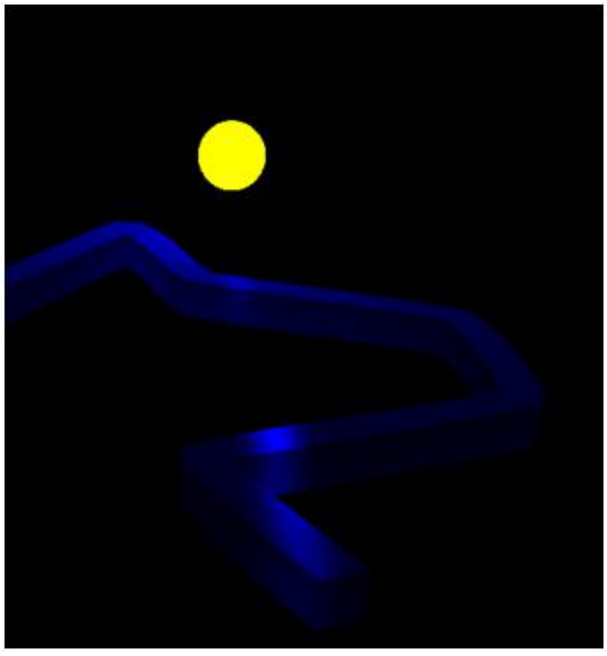


Abbildung 13: Abgeändertes Phong

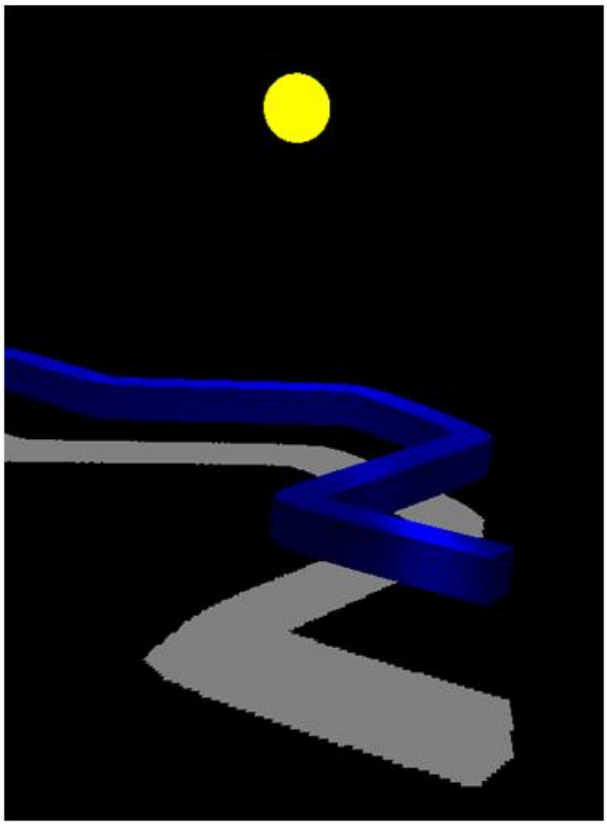


Abbildung 14: Schattenwurf

2.6.5 Probleme, Vorschläge und Erweiterungen

Nachtmodus verbessern:
Bisher bleibt die Beleuchtung bei Nacht (Sonnenwinkel ¡ 20°) in ihrem letzten Zustand. Das Bedeutet die Beleuchtung wird nur eingefroren und nicht wie gewünscht um ihre spiegelnde Reflexion reduziert. Wünschenswert wäre eine gleichmäßige Beleuchtung gänzlich ohne spiegelnden Anteil.

Schattenfehler korrigieren:
Zur Zeit besteht noch ein Fehler beim Erzeugen des Schattens. Bei bestimmten Blickwinkeln entstehen Streifen die sich über das Bild ziehen.

Schatten verfeinern:
Der Schatten ist bisher sehr grobkörnig und pixelig. Lösungsvorschlag durch eine erweiterte Shadowmap-Technik wie PCF oder VSM

Andreas Brust, Thi Dieu Hien Le

2.7 Portierung des Frameworks

2.7.1 Ausgangslage

Das In-Car-Multimedia Labor verfügt an jedem Arbeitsplatz über eine komplette Multimedia-Einheit der Firma Audi beziehungsweise Becker. Dieses System besteht unter anderem aus einem Navigationssystem, dem dazugehörigen Display und einem sogenannten MaxiCommander zum Tätigen der Eingaben.

Für dieses System gibt es ein Framework, das fortlaufend weiterentwickelt wird. Das Framework ist

kompilierbar für Windows (32 Bit) und QNX, einem proprietären Echtzeitbetriebssystem des gleichnamigen Anbieters [QNX10]. Die QNX-Version läuft auf der Audi/Becker Hardware.

2.7.2 Zielsetzung

Um nicht mehr auf die Audi/Becker Hardware angewiesen zu sein, wurde ein Rechner mit Intel Atom Prozessor angeschafft (sog. Nettop), auf dem Linux als Betriebssystem eingesetzt werden soll. Hierzu ist es notwendig, das Framework entsprechend anzupassen und für Linux kompilierbar sowie lauffähig zu machen. Eine weitere wichtige Anforderung ist es, die Kompilierbarkeit des Frameworks für Windows und QNX nicht zu beeinträchtigen. Darüber hinaus ist das Framework in Komponenten (beispielsweise für Ipod, Radio/Tuner, Navigation etc.) aufgebaut, die ebenfalls ohne Ausnahmen für Linux portiert werden sollen.

2.7.3 Umsetzung

Zu den vorbereitenden Arbeiten gehörte es, eine virtuelle Maschine mit Linux (Ubuntu 9.04) aufzusetzen und entsprechend einzurichten. Als Entwicklungsumgebung wurde Eclipse (Galileo) installiert. Zum Zweck der Versionsverwaltung kam in diesem Semester erstmals Perforce zum Einsatz. Eine Beschreibung zum Vorgehen bei der Installation bzw. Einrichtung liefert Tobias Braun im Kapitel zur Grafikportierung 2.4.

Das Kompilieren des Frameworks geschieht anders als meist üblich nicht aus der Entwicklungsumgebung (Eclipse) heraus, sondern mittels Perforce Jam, einem quelloffenen Build-System, ähnlich wie make [Wik10b]. Den vergleichsweise großen Geschwindigkeitsvorteil beim Kompilieren mit jam erkauft man sich mit einem gewissen Komfortverlust beim Debuggen. Da man per Kommandozeile kompiliert, ist es nicht möglich die Fehler verursachende Stelle im Programmcode innerhalb Eclipse direkt anzusteuern. Aus diesem Grund muss man per Hand zu der entsprechenden Datei sowie Zeile im Programmcode navigieren.

Hier die beispielhafte Ausgabe eines Compilerfehlers:

src/gl_hmi/ss/navi/navi/MapControl.cpp:218: Fehler: “glOrthof” wurde in diesem Gültigkeitsbereich nicht definiert.

Als eine der ersten Maßnahmen musste die OpenGL Grafikbibliothek installiert werden. Dies geschah in Absprache mit der Systemgruppe mittels der Mesa 3D-Bibliothek, die auch auf dem Linux-Zielsystem zum Einsatz kommt. Die genauen Details sind der folgenden Grafik zu entnehmen:

Paket	Installierte Version	Neueste Version	Beschreibung
libgl1-mesa-dri	7.4-0ubuntu3	7.4-0ubuntu3.2	A free implementation of the OpenGL API -- DRI modules
libgl1-mesa-glx	7.4-0ubuntu3	7.4-0ubuntu3.2	A free implementation of the OpenGL API -- GLX runtime
mesa-utils	7.4-0ubuntu3	7.4-0ubuntu3.2	Miscellaneous Mesa GL utilities
libgl1-mesa-dev	7.4-0ubuntu3.2	7.4-0ubuntu3.2	A free implementation of the OpenGL API -- GLX development files
libglu1-mesa	7.4-0ubuntu3.2	7.4-0ubuntu3.2	The OpenGL utility library (GLU)
libglu1-mesa-dev	7.4-0ubuntu3.2	7.4-0ubuntu3.2	The OpenGL utility library -- development files
mesa-common-dev	7.4-0ubuntu3.2	7.4-0ubuntu3.2	Developer documentation for Mesa

Abbildung 15: Liste der benötigten OpenGL-Bibliotheken

Im weiteren Verlauf folgte die Abarbeitung der Compilerfehler und somit das Beheben selbiger. Aus Gründen der übersichtlichkeit ist die Protokollierung samt Erläuterungen im Anhang unter “Protokollierung zur Portierung des Frameworks” zu finden B.1.

2.7.4 Anmerkung

Gelegentlich kam es bei der Arbeit mit Perforce bei dem Versuch die aktuellste Version des Frameworks vom Server herunterzuladen zu Fehlermeldungen. Ihr Inhalt: „can’t clobber file...“ mit Angabe eines Dateinamens. Der Fehler war leider nicht reproduzierbar, ließ sich jedoch mit einem Workaround umgehen. Hierbei müssen die Zugriffseigenschaften der betroffenen Dateien im Dateisystem auf “nur lesen” bzw. “read only” gesetzt werden [Ram10].

2.7.5 Fazit

Mittlerweile kompiliert das Framework ohne Probleme für bzw. unter Linux. Es wurden alle Fehler beseitigt. Das Kompilieren ist nun mit Angabe der gewünschten Compilerversion möglich (z.B. “jam -sG++VERSION=4.2” für die Version 4.2 von GNU). Wie gewünscht ist die Kompilierbarkeit für Windows und QNX vollständig erhalten geblieben.

Beim Starten des Frameworks gab es jedoch zunächst noch leichte Komplikationen. Zunächst wurde eine Fehlermeldung mit dem Inhalt “Segmentation Fault” ausgegeben, was sich jedoch beheben ließ. Die nächste Fehlermeldung (can’t find map\tree.dat) ließ sich auf das noch nicht vorhandene Kartenmaterial zurückführen. Hier muss nach dem Kompilieren des Projekts in “bin\linux” der Ordner “map” erstellt werden. Anschließend kopiert man dort das Kartenmaterial aus dem Ordner “Navi_NG3_Navimaps\navimaps_old bzw. navimaps_new” (Perforce, Stammverzeichnis des Projekts) hinein. Anschließend startet das Framework ohne Fehlermeldungen B.1.

Christian Bartel

2.8 Systemgruppe

2.8.1 Implementierung der GPS-Komponente

Zu Beginn des Semesters war das in den vorangegangen Semestern entwickelte Framework unter Linux noch nicht lauffähig. Um neue Komponenten für den USB-GPS-Empfänger und den Minicommander entwickeln zu können, benötigten wir zumindest ein minimales Framework. Von Laboringeneur M.Sc. Andreas Knirsch haben wir hierfür aus seiner Masterarbeit ein solches minimales Framework bekommen. In diesem ließen sich die neu entwickelten Komponenten testen. Das Einbinden der Komponenten in das minimale Framework im Gegensatz zum Haupt-Framework unterscheidet sich lediglich beim Konstruktoraufruf.
Um eine reibungslose Zusammenführung zurück in das alte Framework zu gewährleisten, haben wir die alte GPS-Komponente extrahiert und in das minimale Framework übernommen. Die notwendigen Anpassungen wurden innerhalb dieser Komponenten vorgenommen. Hierbei mussten wir auch die Abhängigkeiten der Komponenten untereinander beachten. Daher wurden beispielsweise auch die GADA-Schnittstelle und diverse Datencontainer übernommen.
Auf dem alten NG3-Target wurden die GPS-Daten in Form von Binärdateien im Dateisystem des Targets zum Auslesen bereitgestellt. Der neue USB-GPS-Empfänger stellt die Daten über die serielle- bzw. USB-Schnittstelle im NMEA-0183-Format zur Verfügung. Die NMEA-0183-Datensätze werden in der neuen GPS-Komponente entsprechend ihrer Beschreibung ausgelesen. Das Framework selbst bekommt diese

Daten über Datencontainer bzw. Messages zur Verfügung gestellt.

Von dem neuen USB-GPS-Empfänger werden insgesamt 13 verschiedene NMEA-0183-Datensätze [uDMW10], als Zeichenketten, über die Gerätedatei (/dev/ttyACM0), zur Verfügung gestellt. Für die von uns bislang gewünschten GPS-Daten (longitude, latitude, year, month, day, utc_hour, utc_minute, utc_second, speed, heading, height), benötigen wir die nach folgenden drei NMEA-0183-Datensätze:

GPS recommended minimum sentence C
$GPRMC,191410,A,4735.5634,N,00739.3538,E,0.0,0.0,181102,0.4,E,A*19
Dabei bedeuten:
- GPRMC: GPS recommended minimum sentence C
- 191410: Uhrzeit der Bestimmung: 19:14:10 (UTC-Zeit)
- A: Status der Bestimmung: A=Active (gültig); V=void (ungültig)
- 4735.5634,N: Breitengrad mit (Vorzeichen)-Richtung (N=Nord, S=Süd) 46 Grad 35.5634' Nord
- 00739.3538,E: Längengrad mit (Vorzeichen)-Richtung (E=Ost, W=West) 007 Grad 39.3538' Ost
- 0.0: Geschwindigkeit über Grund (Knoten)
- 0.0: Bewegungsrichtung in Grad (wahr)
- 181102: Datum der Bestimmung: 18.11.2002
- 0.4,E: Missweisung (mit Richtung)
- A: Neu in NMEA 2.3: Art der Bestimmung (A=autonomous (selbst), D=differential, E=estimated (geschätzt), N=not valid (ungültig), S=simulator)
- *19: Prüfsumme

GPS Global Positioning System Fix Data
$GPGGA,191410,4735.5634,N,00739.3538,E,1,04,4.4,351.5,M,48.0,M,,*45
Dabei bedeuten:
- GPGGA: GPS Global Positioning System Fix Data
- 191410: Uhrzeit der Bestimmung: 19:14:10 (UTC-Zeit)
- 4735.5634, N: Breitengrad
- 00739.3538,E: Längengrad
- 1: Qualität der Messung (0 = ungültig, 1 = GPS, 2 = DGPS, 6 = geschätzt nur NMEA-0183 2.3)
- 04: Anzahl der erfassten Satelliten
- 4.4: HDOP (horizontal dilution of precision) Genauigkeit
- 351.5, M: Höhe über Meer (über Geoid) in Metern (351.5,M)
- 48.0, M: Höhe Geoid minus Höhe Ellipsoid (WGS84) in Metern (48.0,M)
- *45: Prüfsumme

GPS Track Made Good and Ground Speed
$GPVTG,0.0,T,359.6,M,0.0,N,0.0,K*47
Dabei bedeuten:
- GPVTG: GPS Track Made Good and Ground Speed
- 0.0,T: Kurs (wahr, T)
- 359.6,M: Kurs (magnetisch, M)
- 0.0,N: Geschwindigkeit ÃŒber Grund in Knoten (N)
- 0.0,K: Geschwindigkeit ÃŒber Grund in km/h (K)

- *47: Prüfsumme

Zum Testen der Implementierung für den USB-GPS-Empfänger existiert im Perforce ein Projekt *Testapp*. Die Applikation stellt die minimal benötigten Komponenten zur Verfügung und ermöglicht eine Ausgabe der extrahierten GPS-Daten als Konsolenausgabe. Zusätzlich werden entsprechende Meldungen des Frameworks ausgegeben.

Beispiel: Funktionsweise der alten GPS-Komponente

Wie bereits erwähnt, wurden auf dem alten NG3-Target die GPS-Daten in Form von Binärdateien im Dateisystem des Targets zum Auslesen bereitgestellt. Bei der Leseoperation dieser Binärdateien wurde die Struktur *GpsAll* mit den aktuellen Daten aus dem GPS-Empfänger ausgelesen. Zur Vergleich ist ein Codeauszug hier gezeigt. Weitere Details hierzu sind der Dokumentation vom Sommersemester 2009 zu entnehmen.

```
1  struct GpsAll
2  {
3          UInt32  timestamp;
4          UInt8   state;
5          UInt8   flags;
6          UInt8   antennaState;
7          UInt8   signalQuality;
8          Int32   latitude;                    // in WGS84 1/1,000,000's of a degree
9          Int32   longitude;                   // in WGS84 1/1,000,000's of a degree
10         Int32   height;                      // in 0.1 meter units
11         UInt32  speed;                       // in 0.01 kmh units
12         UInt16  heading;                     // in 0.1 degrees
13         UInt16  year;
14         UInt8   month;
15         UInt8   day;
16         UInt8   utc_hour;
17         UInt8   utc_minute;
18         UInt8   utc_second;
19         UInt8   fix;
20         UInt16  hdop;                        // in 0.01 units
21         UInt16  pdop;                        // in 0.01 units
22         UInt16  vdop;                        // in 0.01 units
23         UInt16  sat_used;
24         UInt16  sat_visi;
25         UInt16  hori_pos_err;
26         UInt16  vert_pos_err;
27         UInt16  north_speed;
28         UInt16  east_speed;
29         UInt16  vert_speed;
30 };
31 ...
32 ...
33 // read-operation
34 fread((void *) &gps, sizeof(GpsAll), 1, mAllGps);   // gps is reference to binary file; mAllGps=struct
```

Beispiel: Auslesen der Zeichenkette $GPVTG der neuen GPS-Komponente

In der Methode *parseGPS* werden von den insgesamt 13 verschiedenen NMEA-0183-Datensätze [uDMW10], die als Zeichenketten vorliegen, die Datensätze *GPRMC*, *GPGGA*, *GPVTG*, geparst und die derzeit benötigten Daten extrahiert. Die extrahierten Daten werden mittels der Struktur *GpsInfoData* in den GPS-Datencontainer geschrieben.

```
1  static void parseGPS(const char* msg, GpsInfoData *gid) {
```

```
...
...
        if (pStart != NULL) {
                char* pEnd = strstr(pStart, "\n");
                if (pEnd != NULL && pEnd - pStart < (int) sizeof(buf)) {
                        strncpy(buf, pStart, pEnd - pStart);
                        buf[pEnd - pStart] = '\0';

                        // now tokenize for determining the contained values
                        char token[TOKENMAXSIZE];
                        const char* tosearch = buf; // search pointer
                        const char* nextStart = NULL;

                        size_t counter = 0;
                        while ((nextStart = tokenize(tosearch, token, ',')) != NULL) {
                                switch (counter) {
                                case 0: //$GPVTG$
                                        //not used
                                        break;
                                case 1: //course
                                        //not used
                                        break;
                                case 2: //anything
                                        //not used
                                        break;
                                case 3: //course magnetic
                                        gid->heading = atof(token);
                                        break;
                                case 4: //anything
                                        //not used
                                        break;
                                case 5: //groundspeed knots
                                        //not used
                                        break;
                                case 6: //speed direction
                                        //not used
                                        break;
                                case 7: //groundspeed km/h
                                        gid->speed = (float)atof(token);
                                        break;
                                case 8: // K for Kilometers
                                        //not used
                                        break;
                                default:
                                        DEBUG_TRACE("not able to handle position %d for elemnent %s",
                                             counter, token);
                                        break;
                                } // switch
                                tosearch = nextStart;
                                ++counter;
                        } // while
                } // if
}
```

2.8.2 Implementierung der Minicommander-Komponente

Da sich der Maxicommander des NG3-Targets nicht an dem neuen Target verwenden lässt, musste auf den (älteren) Minicommander, der über eine serielle Schnittstelle angesprochen werden kann, zurück gegriffen werden. Für die Nutzung der Funktionalitäten des Minicommanders wurde ebenfalls eine Framework-Komponente implementiert.

Die Kommandos des Minicommanders werden ebenso, wie die Datensätze des USB-GPS-Empfängers, in einer Gerätedatei (/dev/ttyS0) zur Verfügung gestellt. Die in diese Datei geschriebenen Kommandos werden von der Methode *interpretTelegram()* ausgewertet und über das Messaging-System des Frame-

works, als Events, an die HMI-Komponente gesendet. Bei der Interpretation der Kommandos ist darauf zu achten, dass die in der Dokumentation des Minicommanders angegeben Werte für Bitrate (B57600) und Threshold (2) eingehalten werden. Die Bedeutung der Kommandos kann aus der Dokumentation des Minicommanders entnommen werden.
Zum Testen der Implementierung für den Mincommanders existiert im Perforce ein Projekt *Testapp*. Die Applikation stellt die minimal benötigten Komponenten zur Verfügung und ermöglicht eine Ausgabe der empfangenen Eingaben als Konsolenausgabe. Zusätzlich werden entsprechende Meldungen des Frameworks ausgegeben.

2.8.3 Aufbau und Installation eines selbst-konfiguierten Mini-Linux

Im Anschluss daran wurde mit dem Aufbau eines selbst-konfiguierten Mini-Linux begonnen. Hierfür benötigten wir die Kernel-, die Busybox-, sowie die XServer-Sourcen aus dem Web. [Ker10] [Bus10] [Fre10]
Die Sourcen des Kernels sind unter http://www.kernel.org, die der Busybox unter http://busybox.net/ und die des XServers unter http://www.x.org/ erhältlich.
Der Kernel stellt den Betriebssystemkern dar, die Busybox stellt einfache Programme zur Verfügung, wie cp oder ls und der XServer ist für die grafische Darstellung zuständig.
Der Kernel und die Busybox wurden so konfiguriert, dass sie lediglich die minimal benötigte Funktionalität zur Verfügung stellen. Für das neue Target wird so ein optimales und sehr schlankes Betriebsystem erzeugt. Wiederum heißt das allerdings auch, dass wenn in der Zukunft neue Funktionalitäten benötigt werden, das Betriebssystem ggf. neu konfiguriert und compiliert werden muss. Um diesen Vorgang zu vereinfachen, wurde von uns eine make-Datei erstellt, welche die Sourcen automatisch aus dem Internet herunterläd und anschließend anhand der Konfigurationsdateien neu compiliert. Die erstellte make Datei befindet sich im Perforce Projekt SystemBuild und die Konfigurationsdateien im Unterordner configuration.

Mit dem Befehl *make prepare* werden die Sourcen des Kernels und die der Busybox heruntergeladen und die von uns erstellten Konfigurationsdateien aus dem Ordner *configuration* in die jeweiligen source Ordner kopiert.
Möchte man die Konfiguration des Kernels oder die der Busybox ändern, muss man in den jeweiligen Source Ordner *\source\busybox* oder *\source\kernel* wechseln und mit dem Befehl *make menuconfig* ein Konfigurationsprogramm aufrufen. Das geänderte Konfigurationsfile *.config* sollte man anschließend sichern, da es ansonsten bei einem erneuten *make prepare* in unseren *Systembuild*-Projekt wieder mit unserer Konfiguration überschrieben wird.

Der Befehl *make image* in unserem Makefile ist für das Compilieren der Busybox und des Kernels verantwortlich. Hierbei muss man darauf achten, dass beim Compilieren des Kernels die Datei *initramfs.conf* aus dem Ordner *initramfs* berücksichtigt wird, um das Filesystem unseres neuen Images zu erstellen. In dieser Datei muss jedes Verzeichnis und jede Datei angegeben werden, die beim Starten des Images verfügbar sein soll. Sollen neue Dateien wie z.B. Libraries oder das Framework auf dem neuen Image verfügbar sein, dann muss das hier angegeben werden. Das compilierte Image wird anschließend im Ordner *image* abgelegt.

Das Compilieren des X-Servers benötigt eine ganze Menge externer Programme und hinzu kommt noch, dass das Compilieren an sich meherere Stunden dauert, wurde dieser Ablauf in die externe Datei *buildxserver.sh* ausgelagert. Die Datei *buildxserver.sh* installiert zu Beginn alle für das Compilieren des XServers benötigten Anwendungen. Hierbei ist darauf zu achten, dass es von Ditribution zu Distribution verschieden ist, welche Anwendungen benötigt werden. Unsere Konfiguration ist nur für Ubuntu Karmic Koala geeignet. Möchte man den XServer auf einer anderen Distribution compilieren, dann muss unter http://wiki.x.org/wiki/RequiredPackages nachgesehen werden, welche Applikationen man braucht. Sind die Programme installiert, wird mit git://anongit.freedesktop.org/git/xorg/util/modular ein Git Repository auf den Rechner kopiert. Dieses enthält wiederum ein Buildscript, welches alle noch benötigten Sourcen herunterläd und compiliert.
Um eine zentrale Anlaufstelle beim compilieren des Systems zu haben, kann das *buildscript.sh* auch mit *make xserver* aufgerufen werden.

Damit wir bei unseren Tests das Target nicht ständig neu starten und von einer anderen Partition booten mussten, wurde der Befehl *make virtual* erstellt. Dieser Befehl startet das im Ordner *image* abgelegte Image in *Qemu*. *Qemu* ist eine Virtualisierungssoftware. Diese muss selbstverständlich auf dem System, auf dem dieser Befehl ausgeführt werden soll, installiert sein.

Nach erfolgreicher Konfiguration des Linux-Images und evt. Tests, kann mit dem Befehl *make install* das Image an einem beliebigen Ort installiert werden.

Christian Müller, Marcus von Rohden

3 Ergebnis

Nachfolgend werden die Ergebnisse des vergangenen Semesters von den einzelnen Gruppen zusammengefasst.

Die Ergebnisse haben bisher gezeigt, dass es sich mit vergleichsweise wenig Rechenaufwand leicht realisieren lässt, die Umrisse von Straßen-Verkehrszeichen mit einer angemessenen Genauigkeit erkennen zu können, ohne den Inhalt zu interpretieren. Zum aktuellen Stand der Entwicklung wäre es durchaus möglich, erkannte Verkehrsschilder aus dem Sichtfeld der Kamera zu kopieren und in die Windschutzscheibe einzublenden, ohne dass deren Bedeutung verstanden sein muss.

In der Gimmick-Komponente muss der Algorithmus zur Detektion einer Kurve hinsichtlich komplexer Kurven erweitert werden. Um dies zu ermöglichen müssen Kurven auf Basis von GPS-Daten erkannt und klassifiziert werden. Des Weiteren muss ein funktionstüchtiger Punktestandzähler eingebaut werden, welcher den Punktestand verwaltet. Nachdem diese Spielelemente fertiggestellt sind, kann damit begonnen werden, weitere Spielelemente (z.B. Fahrspurerkennung zur Überprüfung des Rechtsfahrgebotes) in das Spiel zu integrieren und den Ansprüchen des Spiels angepasst zu werden.

Für die Fahrspurerkennung wurde ein Algorithmus implementiert, der ein Video einliest und in diesem Video nach Fahrspurmarkierungen sucht. Die gefundenen Markierungen werden für die weitere Verwendung gespeichert. Wegen Zeitmangels werden die Markierungen jedoch noch nicht als Fahrspuren im System für die Pfeildarstellung genutzt. Zum Testen des Algorithmus wurde eine Testfahrt auf der Autobahn aufgezeichnet, da diese wegen guten Kontrasten, wenigen Schatten und guten Markierungen optimale Bedingungen für die Erkennung bietet.

Für das neue Target wurde ein Embedded Linux mit minimaler Anforderung konfiguiert und installiert. Das Embedded Linux wird skript-gesteuert erzeugt und installiert. Nachträgliche Änderungen bzw. Anpassungen sind daher leicht vorzunehmen. Entsprechend der neuen Hardware-Umgebung wurde ein USB-GPS-Empfänger in Betrieb genommen und für das Framework eine angepasste GPS-Komponente entwickelt. Zur Bedienung der Software steht der Minicommander zur Verfügung. Hierfür wurde ebenfalls eine angepasste Framework-Komponente entwickelt.
Ein abschließender Test mit dem portierten Framework konnte nicht mehr durchgeführt werden. Hierfür sind u.a. auch noch Anpassungen an dem X-Grafiksystem des selbst-konfigurierten Embedded Linux, je nach Anforderungen des portierten Framworks, vorzunehmen.

Bezüglich der Portierung des Frameworks lässt sich ein Erfolg vermelden. Das Framework kompiliert und läuft nun fehlerfrei unter Linux. Dem Einsatz auf dem Zielsystem steht somit nichts mehr im Wege. Darüber hinaus ist das Framework weiterhin für Windows und QNX kompilierbar.

Die Portierung des Navi3D-Clienten konnte erfolgreich umgesetzt werden. Die Schnittstelle zur SQL-Datenbank muss überarbeitet werden, da diese in C# entwickelt wurde und so nicht einfach unter Linux lauffähig war. Diese Aufgabe wurde aber anderweitig vergeben und gehörte somit nicht mehr zur Zuständigkeit der Portierungsgruppe. Da der Navi3D-Client nun auch direkt über eine XML-Datei konfiguriert werden kann, ist es sehr einfach verschiedene Zenarien auf dem neuen Target zu konfigurieren und zu testen. Dies erfordert nun kein erneutes Kompilieren des Navi3D-Clienten mehr.

4 Ausblick

Zur Auswertung der bisher ausgeschnittenen Verkehrsschilder soll in Zukunft ein neuronales Netz dienen, welches trainiert und getestet wird. Für die Weiterführung des Projektes ist es durchaus denkbar, dass auf längere Sicht gesehen genug Bildmaterial verfügbar sein wird, um eine neuronales Netz zu trainieren. Darüber hinaus wird sich erst bei der Implementierung des neuronalen Netzes herausstellen, ob unser Ansatz zur computergestützten Erkennung von Straßen-Verkehrsschildern echtzeitfähig und gleichzeitig zuverlässig ist.

Für eine optimale Nutzung der Fahrspurerkennung wäre es wünschenswert, wenn die erkannten Markierungen für die Pfeildarstellung genutzt werden. Zusätzlich müssten noch bestehende Probleme bei der Interpretation der Markierung als Fahrspur gelöst werden. Hierbei spielen teilweise variierende Markierungen eine Rolle, wie z.B. durchgezogene oder gestrichelte Linien sowie fehlende Markierungen. Außerdem könnten die Markierungen anhand ihrer Farbe unterschieden werden, da beispielsweise gelbe Baustellenmarkierungen einen Vorrang vor den “normalen” Fahrspuren haben. Zusätzlich müsste noch eine Unterscheidung zwischen der eigenen und fremden Fahrspur erfolgen. Die bereits im Hauptteil erwähnten Probleme wie Schatten, Licht, Schnee usw. bedürfen weiterer Aufmerksamkeit.

Im Bereich der Entwicklung der Lernspiele steht die Kapselung einzelner Spielelemente an. Die Klasse Navi3D_Gimmick stellt einen einfachen Weg dar, bisher bestehende Verfahren zur Überprüfung von verkehrssicherem Fahren, wie bspw. der Fahrspur- oder Schilderkennung in das Projekt bzw. das Lernspiel zu integrieren. Ziel ist die Kapselung einzelner Spielelemente in einer modular aufgebauten Struktur, welche in verschiedenen Spielmodi wiederverwendet werden können.

Bezüglich den Portierungen des Navi3D-Client und des Frameworks ergeben sich ebenfalls Anschlussarbeiten. Der Navi3D-Client steht für die Nutzung auf dem neuen Target bereit. Da das neue Targetsystem noch nicht ganz lauffähig war, konnte ein abschließender Test auf dem System nicht durchgeführt werden. Als nächstes muss also der Navi3D-Client mit allen Komponenten zusammen auf den neuen Target getestet werden. Dies sollte aber kein Problem sein, da er auf verschiedenen Linuxdistributionen (Open Suse und Ubuntu) erfolgreich getestet wurde. Des weiteren muss die SQL-Schnittstelle wieder integiert werden, damit die POIs wieder aktiviert werden können. Auch die Kommunikation mit dem Framework muss neu geschrieben werden. Dies resultiert zum einen daraus das die Fähigkeiten der Gadaschnittstelle nicht den Anforderungen einer ausfallsicheren Kommunikation genügen. Zum anderen wird das Framework und der Navi3D-Client zusammen auf dem neuen Target laufen. Somit ist eine Netzwerk basierte Kommunikation hier nicht mehr nötig. Hierfür kann zum Beispiel auf Message Parsing zurückgegriffen werden. Die GADA-Schnittstelle selbst kann weiterhin für die Kommunikation mit der Beckerdemo genutzt werden. Dies kann zusammen mit einer integration in das Framework als Modul umgesetzt werden.

Auch das Framework ist nun unter Linux lauffähig und wurde diesbezüglich ebenfalls auf den Distibutionen Open Suse und Ubuntu getestet. Als nächster Schritt steht der produktive Einsatz auf dem Zielsystem bevor, an dem noch kleine Anpassungen notwendig waren.

Im Bereich der Systemumstellung sind noch Anpassungen an dem X-Grafiksystem des selbst-konfigurierten Embedded Linux vorzunehmen. Hierfür müssen zunächst die Anforderungen des portierten Framework identifiziert werden und das System entsprechend angepasst werden.

Anhang

A Grafik - Codedokumentation

A.1 Verkehrsschilderkennung

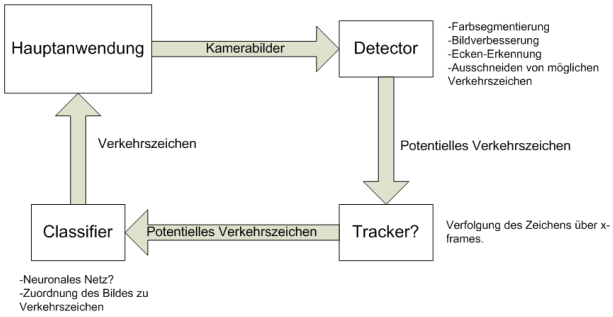


Abbildung 16: Architektur Verkehrsschilderkennung

Die Implementierung orientierte sich stark an dem Diagramm aus Abbildung 16, wobei bisher nur zwei der vier Klassen realisiert wurden. Die Klasse *Navi3D* stellt die Schnittstelle zum Hauptprojekt „Navi3D“ dar:

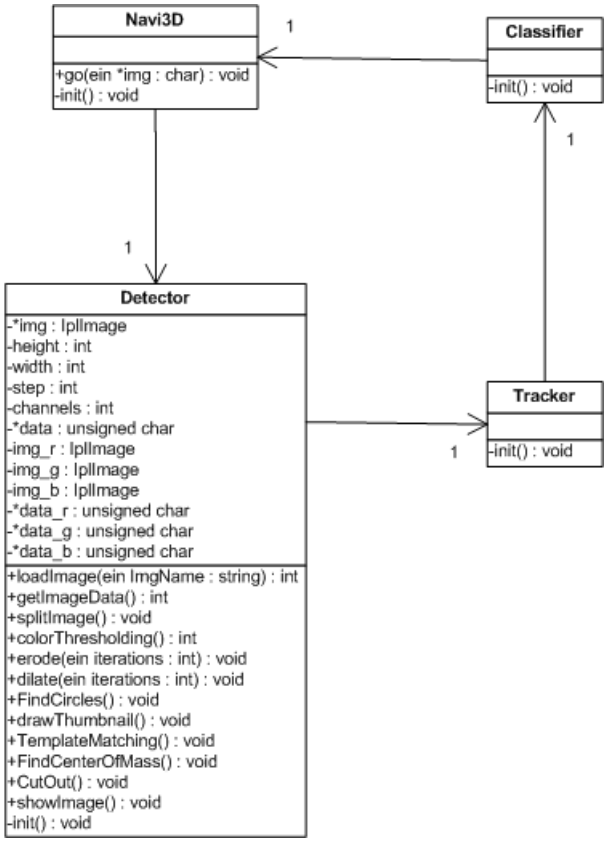


Abbildung 17: Klassendiagramm Verkehrsschilderkennung

Im Folgenden werden die relevanten Methoden *splitImage()*, *getImageData()*, *colorThresholding()*, *FindCircles()* und *drawThumbnails()* der Klasse *Detector* näher beleuchtet.

Die Methode *splitImage()* wird ausgeführt, nachdem das Bild mittels *loadImage(std::string ImgName)* geladen wurde. Sie kopiert das Bild in den Speicher von *img_r*, *img_g*, *img_b*, damit nach dem Color-

Thresholding jeweils ein Bild für die drei r, g und b Kanäle verwendet werden können, was einen direkten Zugriff auf die gefilterten Farben erlaubt.

Listing 5: Detector.cpp - splitImage()

```
1  void Detector::splitImage(){
2          img_r = cvCreateImage( cvGetSize(img), IPL_DEPTH_8U, 3 );
3          img_g = cvCreateImage( cvGetSize(img), IPL_DEPTH_8U, 3 );
4          img_b = cvCreateImage( cvGetSize(img), IPL_DEPTH_8U, 3 );
5  }
```

In der Methode *colorThresholding* wird das in Abschnitt 2.1 beschriebene Verfahren verwendet, um das Ursprungsbild in die für die weiteren Berechnungen benötigten Farben aufzuteilen. Hierfür muss jedes Pixel des Ursprungsbildes auf seine RGB Werte überprüft werden. Falls der gewünschte Rot-Anteil in einem Pixel festgestellt wird, kann in dem dazugehörigen *Rot-Bild* das Pixel auf weiß gesetzt werden. Da die Bilder für die unterschiedlichen Farb-Kanäle nach der Initialisierung schwarz sind entstehen somit binäre Bilder, bei denen die weißen Pixel stellvertretend für die jeweilige Farbe im Ursprungsbild stehen.

Aus Platzgründen wird hier nur der Abschnitt für die Erkennung der Farbe rot dargestellt, da das Verfahren für die weiteren Farben vergleichbar arbeitet.

Listing 6: Detector.cpp - colorThresholding()

```
1  int Detector::colorThresholding(){
2          for(int i=0;i<height;i++)
3          {
4                  for(int j=0;j<width;j++)
5                  {
6                          if(data[i*step+j*channels+2] >= 130 &&
7                                  data[i*step+j*channels+1] <= 100 &&
8                                  data[i*step+j*channels+0] <= 100)
9                          {
10                                 data_r[i*step+j*channels+0] = 255; //blue
11                                 data_r[i*step+j*channels+1] = 255; //green
12                                 data_r[i*step+j*channels+2] = 255; //red
13                         }
14                         ...
15                         ...
16                 }
17         }
```

Für die Erkennung der runden Verkehrsschilder wurde die Funktion *FindCircles()* erstellt. Zunächst muss hierfür beispielsweise das Bild mit den roten Farbwerten (*img_r*) in ein Graustufenbild gewandelt werden, damit mittels der OpenCV-Funktion *cvHoughCircles(.....)* die roten Kreise innerhalb des Bildes erkannt und die Parameter der Kreise (x-Koordinate, y-Koordinate, Radius) in die Struktur *circles* gespeichert werden können. Anschließend werden die Kreise auf das Ursprungsbild gezeichnet indem durch die Elemente von *circles* iteriert und mit den Parametern der Kreise die OpenCV-Funktion *cvCircle(.....)* aufgerufen wird. Die Variable *correctFactorRad* dient in der Zeile 16 der Vergrößerung der Radien, da *cvHoughCircles(.....)* den inneren Kreis des Verkehrszeichens erkennt. Um jedoch das Verkehrszeichen vollständig zu umranden, muss der Radius relativ zur Größe des Schildes vergrößert werden.

Listing 7: Detector.cpp - FindCircles()

```
1  void Detector::FindCircles(){
2          IplImage* gray = cvCreateImage(cvGetSize(img_r), 8, 1);
3          CvMemStorage* storage = cvCreateMemStorage(0);
4          cvCvtColor(img_r, gray, CV_BGR2GRAY);
5          cvSmooth(gray, gray, CV_GAUSSIAN, 9, 9);
6          circles = cvHoughCircles(gray, storage, CV_HOUGH_GRADIENT, 2, gray->height/20, 300, 100);
7
8          int signX, signY, signRad = 0;
```

```
9           float correctFactorRad = 1.6;
10
11          for (int i = 0; i < circles->total; i++)
12          {
13               float* p = (float*)cvGetSeqElem( circles, i );
14               signX = p[0];
15               signY = p[1];
16               signRad = cvRound(correctFactorRad * p[2]);
17
18                   //draw circle
19                   cvCircle( img, cvPoint(signX, signY), signRad, CV_RGB(0,255,0), 4, 8, 0 );
20          }
21  }
```

Damit die Verkehrsschilder dem Fahrer in die Windschutzscheibe eingeblendet werden können, müssen die erkannten Verkehrsschilder aus dem Ursprungsbild kopiert und skaliert werden. Hierfür dient die Methode *drawThumbnails()*, die zunächst in der Zeile 10 ein temporäres Bild *placeholderSign* mit der Größe 90 auf 90 Pixel erstellt.

Für die Kopie des Verkehrszeichens iteriert die Methode durch *circles* und erstellt in Zeile 22 ein Quadrat um das Verkehrszeichen. Dieser quadratische Ausschnitt des Bildes wird in den folgenden Schritten auf dem Ausgabebild platziert. Dazu ist es notwendig vorher eine sogenannte *Region of Interest (ROI)* festzulegen. Diese ROI spezifiziert einen Teil des Bildes, welcher gerade zur weiteren Bearbeitung genutzt wird.

Listing 8: Detector.cpp - drawThumbnails()

```
1   void Detector::drawThumbnail(){
2           IplImage* img_re = 0;
3           int signX, signY, signRad = 0;
4           int tmpImgLength = 75;
5           float correctFactorRad = 1.6;
6           float correctFactorImg = 2.2;
7           int cWidthInPicture = 0;
8           int signLength = 90;
9           float scaleFactor = 1.0;
10          IplImage* placeholderSign = cvCreateImage(cvSize( signLength, signLength ), IPL_DEPTH_8U, 3);
11
12          for (int i = 0; i < circles->total; i++)
13          {
14                  float* p = (float*)cvGetSeqElem( circles, i );
15                  // Werte der Kreiserkennung
16                  signX = p[0];
17                  signY = p[1];
18                  signRad = cvRound(correctFactorRad * p[2]);
19                  tmpImgLength = cvRound(correctFactorImg * signRad);
20
21                  //Rechteck des Bildausschnitts
22                  CvRect rect = cvRect(signX-signRad, signY-signRad, tmpImgLength, tmpImgLength);
23                  //temp Bild
24                  img_re = cvCreateImage(cvSize( tmpImgLength, tmpImgLength ), IPL_DEPTH_8U, 3);
25                  //rechteck auf bild setzen
26                  cvSetImageROI(img, rect);
27                  //erkanntes schild in img_re kopieren
28                  cvCopy(img, img_re, 0);
29                  //roi zuruecksetzen
30                  cvResetImageROI(img);
31                  //skalierung berechnen
32                  scaleFactor = ((float)signLength)/tmpImgLength;
33                  cvResize(img_re, placeholderSign, CV_INTER_CUBIC);
34                  //neues rechteck fuer schild in orig bild
35                  rect = cvRect(cWidthInPicture, img->height-placeholderSign->height, placeholderSign->
                        width, placeholderSign->height);
36                  //rechteck auf ausgangsbild
37                  cvSetImageROI(img, rect);
```

```
38                  //erkannte schild auf orig bild
39                  cvCopy(placeholderSign, img, 0);
40                  cvResetImageROI(img);
41                  //spacer fuers naechste zeichen
42                  cWidthInPicture += signLength;
43          }
44  }
```

Die hier besprochene prototypische Implementierung ist noch verbesserungswürdig und weist in manchen Bereichen Lücken auf, wie z.B. das Fehlen der Auswertung von MPEG-Videos. Diese Lücken werden in dem kommenden Semester zeitgleich mit der Implementierung der Klassen *Tracker* und *Classifier* geschlossen. Darüber hinaus werden derzeit vorhandene Schönheitsfehler behoben. So ist es momentan noch möglich das Programm abstürzen zu lassen, wenn in einem Bild zu viele Verkehrszeichen erkannt und dargestellt werden. Eine Möglichkeit wäre, die Zeichen kleiner und somit in eine Zeile passend zu skalieren. Eine andere ist es, die Zeichen genau einmal anzuzeigen, was aber einem weiteren Fortschritt der Entwicklung bedarf, da die erkannten Schilder so zwangsläufig ausgewertet werden müssen.

A.2 Gimmick

Im Folgenden wird der Code der Klasse „Navi3D_GIMMICK“ erläutert. Die Klasse beinhaltet folgende Funktionalität:

- Detektion von Kurven und Überprüfung, ob eine Kurve mit angemessener Geschwindigkeit befahren wird.
- Einführung einer Geschwindigkeitsvariable, einschließlich Geschwindigkeitsregulation bzgl. der Becker-Demo.

Des Weiteren bedient sich die Klasse „Navi3D_GIMMICK“ bestehender Methoden der Klasse „Navi3D_Arrow“. Hierzu gehören u.a. folgende mathematische Methoden aus dem Bereich der Vektorrechnung: *getVector(), getLength(), normalize(), dotProduct(), getAngle().* Auf diese Methoden wird im Folgenden nicht näher eingegangen, da sie im Vergleich zu den Methoden der Klasse „Navi3D_Arrow“ ggf. nur geringfügig optimiert wurden. Weiterhin wird darauf verzichtet jegliche Set- und Get-Methoden zu beschreiben, da ihre Funktionalität selbsterklärend ist.
Die Klasse „Navi3D_GIMMICK“ hat den folgenden Konstruktor:

Listing 9: Navi3D_GIMMICK.cpp - Konstruktor

```
1  Navi3D_GIMMICK::Navi3D_GIMMICK(Navi3D_Base *tBase, IGADA *tGADA)
2  {
3          geschwindigkeit = 50;           // Geschwindigkeit der Fahrt in der Becker-Demo.
4          pBase = tBase;
5          pGADA = tGADA;
6          warning = false;        // wird true, wenn die für eine Kurve zulässige Geschwindigkeit
               überschritten wird.
7          score = 0;              // der Punktestand des Spiels.
8          curve.x = 0;            // Initialisierung des Kurven-Vertex, welches eine Warnmeldung auslöst
               , mit dem Wert 0.
9          curve.y = 0;
10         curve.z = 0;
11 }
```

Es wird ein Objekt der Klasse „Navi3D_GIMMICK“ in der Datei bzw. im Konstruktor der Klasse „Navi3D_Display.cpp“ angelegt. Zum diesem Objekt bestehen zwei Pointer, welche zum einen als Attribut in der Klasse „Navi3D_Display.h“ und als globale Variable in der Datei „Navi3D_Display.cpp“ angelegt wurden, um eine Interaktion mit diversen Tasten zu ermöglichen.

Durch die beiden Methoden *erhoeheGeschwindigkeit* und *verringereGeschwindigkeit* wird das Attribut *geschwindigkeit* der Klasse um einen festen Wert erhöht bzw. verringert. Hierbei wird verhindert, dass der Wert des Attributs größer als 200 oder negativ wird, da negative Geschwindigkeiten unzulässig sind.

Listing 10: Navi3D_GIMMICK.cpp - erhoeheGeschwindigkeit(int); verringereGeschwindigkeit(int)

```
1  void Navi3D_GIMMICK::erhoeheGeschwindigkeit(int g)
2  {
3          geschwindigkeit += g;
4
5          // 200 MAX GESCHWINDIGKEIT
6          if(geschwindigkeit > 200) geschwindigkeit = 200;
7  }
8
9  void Navi3D_GIMMICK::verringereGeschwindigkeit(int g)
10 {
11         geschwindigkeit -= g;
12
13         // keine negative Geschwindigkeit
14         if(geschwindigkeit < 0) geschwindigkeit = 0;
15 }
```

Die beiden Methoden werden in der Funktion *keyboard* in der Datei „Navi3D_Display.cpp“ aufgerufen. Hierdurch wird ermöglicht, dass in der Becker-Demo durch Drücken der Tasten „e“ bzw. „d“ die Geschwindigkeit der Demo erhöht bzw. verringert wird.

Die Methode *isCurveDangerous* prüft, ob der Winkel, welcher eine einfache bzw. den Anfang einer komplexen Kurve beschreiben soll, zu groß und somit zu scharf ist, um mit der aktuellen Geschwindigkeit befahren zu werden. Ist die Kurve zu scharf, gibt die Methode den bool-wert true zurück. Der Winkel, welcher an die Methode übergeben wird, liegt zwischen den den Vektoren, welche aus drei aufeinanderfolgenden Navigations- bzw. GPS-Koordinatenpunkten der geplanten Route gebildet werden. Der Übergabeparameter *geschwindigkeit* ist optional und wird beim funktionsaufruf mit nur einem Übergabeparameter durch das Attribut *geschwindigkeit* ersetzt.

Listing 11: Navi3D_GIMMICK.cpp - isCurveDangerous(int int)

```
1  bool Navi3D_GIMMICK::isCurveDangerous(int angle, int geschwindigkeit)
2  {
3          bool danger = false;
4
5          if((angle > 135  && geschwindigkeit > 20) ||
6             (angle > 80 && geschwindigkeit > 40) ||
7             (angle > 30 && geschwindigkeit > 100) ||
8             (angle > 20 && geschwindigkeit > 150)) danger = true;
9          else danger = false;
10
11         return danger;
12 }
```

Die Methode *getSpeedInKMH* dient der Umwandlung der aktuellen Geschwindigkeit (int) in einen String. Dieser String kann in der Becker-Demo mittels OpenGl-Funktion ausgegeben werden. Die Methode *getSpeedInKMH* stimmt in ihrer groben Funktionalität mit den beiden Methoden *toString* und *getScore* überein, besitzt jedoch eine Textausgabe (bspw. „km/h“), welche auf die Eingabevariable (die Geschwindigkeit) angepasst wurde.

Listing 12: Navi3D_GIMMICK.cpp - getSpeedInKMH(); toString(int); getScore()

```
1  char* Navi3D_GIMMICK::getSpeedInKMH()
2  {
3          // "speed" to int an write to stringstream
4          stringstream tmp;
5          tmp << static_cast<int>(geschwindigkeit);
```

```
6
7           // write output string
8           string speed_String;
9           tmp >> speed_String;
10          speed_String+=" km/h";
11          speed_String = "Speed: "+speed_String;
12
13          // cast string to char*
14          char *stringOutput;
15          stringOutput = new char[speed_String.length() + 1];
16          strcpy(stringOutput, speed_String.c_str());
17
18          return stringOutput;
19  }
20
21  char* Navi3D_GIMMICK::toString(int x) // (graphischer) DEBUG
22  {
23          // write to stringstream
24          stringstream tmp;
25          tmp << static_cast<int>(x);
26
27          // write output string
28          string temp_String;
29          tmp >> temp_String;
30
31          char *stringOutput;
32          stringOutput = new char[temp_String.length() + 1];
33          strcpy(stringOutput, temp_String.c_str());
34
35          return stringOutput;
36  }
37
38  char* Navi3D_GIMMICK::getScore()
39  {
40                  // "speed" to int an write to stringstream
41          stringstream tmp;
42          tmp << static_cast<int>(score);
43
44          // write output string
45          string speed_String;
46          tmp >> speed_String;
47          speed_String+=" points";
48          speed_String = "Score: "+speed_String;
49
50          // cast string to char*
51          char *stringOutput;
52          stringOutput = new char[speed_String.length() + 1];
53          strcpy(stringOutput, speed_String.c_str());
54
55          return stringOutput;
56  }
```

Die Methode *getSpeedInKMH* wird als Übergabeparameter an die Funktion *renderSpacedBitmapString* in der Methode *Geometry* der Klasse „Navi3D_Display“ übergeben und ermöglicht die Darstellung der aktuell gefahrenen Geschwindigkeit in der Becker-Demo. Unmittelbar danach wird die Funktion *renderSpacedBitmapString* mit einer Warnmeldung aufgerufen, sofern zuvor die Methode *getWarning* den Wert true zurückgibt, also folglich eine Warnmeldung ausgegeben werden muss.

Listing 13: Navi3D_GIMMICK.cpp - getSpeedInKMH(); toString(int); getScore()

```
1  renderSpacedBitmapString((winwidth/2)-150,winheight-150,10,(void *)font,pGIMMICK->getSpeedInKMH()); //
       GIMMICK: speed in km/h (gltext)
2  if(pGIMMICK->getWarning())
3          renderSpacedBitmapString((winwidth/2)-150,winheight-200,10,(void *)font,"slow down"); //
               GIMMICK: warning text
```

Die Methode *calculateRouteData* besitzt die Aufgabe zu überprüfen, ob aktuelle die Becker-Demo oder das Target angesprochen werden und in Abhängigkeit davon die Methode *calculateWarnings* mit dem Geschwindigkeitswert der Demo oder vom GADA-Interface aufzurufen.

Listing 14: Navi3D_GIMMICK.cpp - calculateRouteData()

```
1  void Navi3D_GIMMICK::calculateRouteData()
2  {
3          // for Speed Control
4          #ifdef BECKER_MODE
5                  calculateWarnings(geschwindigkeit);
6          #else
7                  calculateWarnings(pGADA->getSpeed());
8          #endif
9  }
```

Die Methode wird in der Funktion *display* der Datei „Navi3D_Display.cpp“ aufgerufen.

Die Methode *calculateWarnings* ruft bisher nur lediglich die Methode *getCurves* auf. Es ist angedacht, dass in der Methode *calculateWarnings* zu späterem Zeitpunkt eine komplexere Kurvenüberprüfung durchgeführt werden soll.
In der Methode *getCurves* werden sämtliche in der aktuellen Sekunde vorhandenen Streckenpunkte (Vertices) nacheinander durchlaufen. Hierbei wird für jeden Vertex mithilfe dessen Vorgänger- und Nachfolger-Vertex der aktuelle Winkel zwischen den zwei Vektoren, welche aus den drei Streckenpunkten resultieren, berechnet. Um den Winkel unabhängig von seiner Orientierung zu erhalten, wird der berechnete Winkel mit -1 multipliziert. Daraufhin wird geprüft, ob der nun stets positive Winkel den Schwellwert von 20 grad überschreitet. Ist dies der Fall, so liegt eine Kurve vor und es wird im Folgenden mithilfe der Methode *isCurveDangerous* geprüft, ob die aktuelle Geschwindigkeit für die aktuelle Kurve zu hoch ist. Ist die auch der Fall, wird das bool-Attribut *warning* auf true gesetzt und das aktuelle Vertex, welches die Warnmeldung auslöst im Attribut *curve* abgelegt.

Listing 15: Navi3D_GIMMICK.cpp - calculateWarnings(int); getCurves()

```
1  void Navi3D_GIMMICK::calculateWarnings(int speed)
2  {
3          getCurves(); // TODO: funktion soll kurvenbeginn zurückgeben
4          // TODO: hier soll geprüft werden, ob die kurve gefährlich ist.
5  }
6
7  void Navi3D_GIMMICK::getCurves()
8  {
9          // alle Vertices des aktuellen Abschnitts durchlaufen
10         for(int i=0; i < pGADA->verticesCount()-1; i++)
11         {
12                 if(i==0 || i==(pGADA->verticesCount()-1)){} // nichts machen
13                 else
14                 {
15                         // Winkel für alle Vertices des aktuellen Streckenabschnitts berechnen
16                         float angle = (float) getAngle(pGADA->getVertex(i-1), pGADA->getVertex(i),
                               pGADA->getVertex(i+1));
17
18                         float real_curve_angle = 180 - angle;
19
20                         // um immer einen positiven Winkel zu erhalten
21                         if(real_curve_angle < 0) real_curve_angle *= -1; // Rechtskurve
22                         // else // Linkskurve
23
24                         if(real_curve_angle > 20)
25                         {
26                                 curvenwert = real_curve_angle;
27                                 if(isCurveDangerous(curvenwert)) warning = true;
28                                 curve = pGADA->getVertex(i);
```

```
29                                  ///////////////
30                                  // hier muss zu Testzwecken noch ein Bild/ die Bilder mit der
                                        Kurvenwarnung geladen werden.
31                                  // anschließend kann der code für das einblenden der warngraphik in
                                        der Display.cpp unter
32                                  // der Bedingung warning=true eingefügt werden.
33                                  ///////////////
34
35                                  // cout << "Kurven-WINKEL: " << curvenwert << endl;
36                                  break;
37                          }
38                          else
39                          {
40                                  // nach der Kurve
41                                  warning = false;
42                          }
43                  }
44          }
45  }
```

Die folgende Methode *setIntervallForCurve* soll detektieren, ob sich die Kurve über mehrere Streckenpunkte erstreckt. Hierzu wird geprüft, ob der Folgewinkel größer als der aktuelle Winkel der Kurve ist. Ist dies der Fall, so ist die Kurve noch nicht zuende und es wird weiteriteriert. Andernfalls kann davon ausgegangen werden, dass die Kurve über ihren Scheitelpunkt hinweg ist und somit die Kurve zuende geht. Während der Iteration wird mithilfe eines Zählers gemerkt, wieviele Streckenpunkte die Kurve umfasst, bis sie ihren Scheitelpunkt erreicht. Die Funktion ist noch unausgereift und bedarf weiterer Entwicklungsarbeit, sodass sie im aktuellen Projekt noch keinerlei Verwendung findet.

Listing 16: Navi3D_GIMMICK.cpp - setIntervallForCurve(int)

```
1   // gibt bool-wert "true" zurück, falls sich die kurve über mehrere abschnitte erstreckt
2   bool Navi3D_GIMMICK::setIntervallForCurve(int i) // ist ist der zähler der schleife, die über die
        vertices indeces läuft (also: i = index)
3   {
4           bool kurve_zuende = false;
5           int abbruchzaehler = 0;
6           double aktueller_winkel = 0.0; // um zu prüfen, ob der aktuelle winkel größer ist als der
                folgende winkel
7           double folgewinkel = 0.0;
8
9           while(!kurve_zuende && abbruchzaehler<500)
10          {
11                  // sicherheitsabfrage um überlauf zu verhindern FEHLT!!!!!!!!!!
12                  aktueller_winkel = getAngle(pGADA->getVertex(i-1), pGADA->getVertex(i), pGADA->
                        getVertex(i+1));
13                  // sicherheitsabfrage um überlauf zu verhindern FEHLT!!!!!!!!!!
14                  folgewinkel = getAngle(pGADA->getVertex(i), pGADA->getVertex(i+1), pGADA->getVertex(i
                        +2));
15
16                  if(aktueller_winkel >= folgewinkel) // wenn der folgende winkel größer ist als der
                        aktuelle, ist anzunehmen, dass die kurve über den scheitelpunkt hinaus ist.
17                  {
18                          kurve_zuende = true;
19                  }
20
21                  // um endlosloop zu verhindern
22                  abbruchzaehler++;
23          }
24          return 0;
25  }
```

Ebenso unausgereift ist die Methode *setCurveIndices*, weshalb an dieser Stelle nicht näher auf die Funktionalität der Methode eingegangen wird und zudem im aktuellen Projekt nicht aufgerufen wird.

A.3 Fahrspurerkennung

In diesem Kapitel wird der Code der Fahrspurerkennung-Gruppe dokumentiert. Nähere Informationen bietet Abschnitt 2.3.

Die Klasse „Navi3D_ Fahrspurerkennung.cpp“ enthält zwei Methoden, den Konstruktor und die Funktion „detection(void)“.

Listing 17: Navi3D_ Fahrspurerkennung.cpp - Konstruktor

```
1  Navi3D_Fahrspur::Navi3D_Fahrspur(void){
2          global_id = 0;
3          // Open file from Testdrive
4          capture = cvCaptureFromAVI(".\\..\\..\\..\\Navi_NG3_Demofiles\\Video\\03_Autobahn_Schoefferstr
               .avi");
5          img = 0;
6          cvGrabFrame(capture);          // retrieve the captured frame
7          img=cvRetrieveFrame(capture);
8          // Binarized image
9          grayImg = cvCreateImage (cvGetSize (img), img->depth, 1);
10         // Gauss-Filter image
11         gauss = cvCreateImage (cvGetSize (img), img->depth, 1);
12         storage = cvCreateMemStorage(0);
13         // count lines that are found in image
14         lines = 0;
15         // insert in list
16         insert = true;
17 }
```

Im Konstruktor wird das Video eingelesen, sowie die einzelnen Bilder zum Zwischenspeichern der Ergebnisse erstellt. In der Methode „detection(void)“ werden die Informationen aus dem Video verarbeitet. Diese wird von der Methode „Navi3D_ Display.cpp“ für jeden Frame aufgerufen.

Hier wird zunächst ein neues Bild aus dem Video geladen. Danach wird eine so genannte „Region of Interest“ erstellt, d.h. das Bild wird oben und unten abgeschnitten, um irrelevante Bildanteile auszusortieren. Danach wird das Bild binarisiert und mithilfe der Gauss-Funktion geglättet. Danach wird mit dem Canny-Edge-Algorithmus die Kanten im Bild ermittelt, und in diesem Bild erneut eine Region of Interest bestimmt. Danach folgt die eigentliche Fahrspurerkennung, in dem durch die cvHoughLines2-Funktion von OpenCV die Linien im Bild identifiziert werden. Die Funktion speichert die gefundenen Linien in einer Liste, die darauf hin durchlaufen wird. Dabei wird überprüft, ob die Linie den richtigen Winkel hat, um eine Fahrspur zu sein (sie näheres dazu in Abschnitt 2.3). In der Liste „lanes“ werden die Fahrspuren gespeichert. Sobald eine neue Linie gefunden wird, überprüft der Algorithmus, ob dieselbe Linie bereits gefunden wurde. Dazu werden die Werte von rho und theta von der neuen Linie und den „Fahrspuren“ aus der Liste verglichen. Unterscheiden sie sich nur geringfügig, so werden die Parameter der Fahrspur angepasst aber keine neue Linie zu den Fahrspuren hinzugefügt. Ist die Linie hingegen völlig neu und wurde als Fahrspur identifiziert, so wird sie in die Liste eingefügt.

Listing 18: Navi3D_ Fahrspurerkennung.cpp - detection()

```
1  void Navi3D_Fahrspur::detection(void){
2
3          cvGrabFrame(capture);       // retrieve the captured frame
4          img=cvRetrieveFrame(capture);
5
6          // set region of interest on img
7          cvSetImageROI(img, cvRect(0, (int)(img->height*0.3), img->width, (int)(img->height*0.45)));
8
9          cropped_color = cvCreateImage(cvGetSize(img), img->depth, img->nChannels);
10         cvCopy(img, cropped_color,0);
11         cvResetImageROI(img);
12
```

```
13          // binarize and smooth with gauss
14          cvCvtColor( img, grayImg, CV_BGR2GRAY );
15          cvSmooth ( grayImg , gauss , CV_GAUSSIAN , 5, 5);
16          // do canny edge detection
17          canny = cvCreateImage ( cvGetSize (img), img ->depth , 1);
18          cvCanny (gauss , canny , 100, 180) ;
19
20          // set region of interest on canny
21          cvSetImageROI(canny, cvRect(0, (int)(img->height*0.3), img->width, (int)(img->height*0.45)));
22          cropped = cvCreateImage( cvGetSize (canny), canny->depth, 1);
23          cvCopy(canny, cropped);
24          cvResetImageROI(canny);
25
26          // find lines with HoughLines algorithm
27          lines = cvHoughLines2(cropped, storage, CV_HOUGH_STANDARD, 1 ,CV_PI/180, 80);
28    for(int i = 0; i < MIN(lines->total,100); i++ )
29      {
30                  line = (float*)cvGetSeqElem(lines,i);
31                  // arguments of HoughLine
32          rho = line[0];
33          theta = line[1];
34          a = cos(theta);
35                  b = sin(theta);
36          x0 = a*rho;
37                  y0 = b*rho;
38          pt1.x = cvRound(x0 + 1000*(-b));
39          pt1.y = cvRound(y0 + 1000*(a));
40          pt2.x = cvRound(x0 - 1000*(-b));
41          pt2.y = cvRound(y0 - 1000*(a));
42
43                  tmp.rho = rho;
44                  tmp.theta = theta;
45                  tmp.pt1 = pt1;
46                  tmp.pt2 = pt2;
47                  tmp.id = global_id;
48                  tmp.counter = 0;
49                  tmp.erased = false;
50
51                  // check if line is lane
52                  if((theta < (75*PI)/180 && theta > (5*PI)/180 || (theta > (115*PI)/180 && theta <
                        (175*PI)/180))){
53                      it = lanes.begin();
54                      while (it != lanes.end()) {
55                              if(!it->erased){
56                                      // check if line is new, if not change parameters of similar
                                            lane
57                                      if(fabs(it->theta-theta) < 0.33 && fabs(it->rho-rho) < img->
                                            width/8){
58                                          insert = false;
59                                          it->pt1 = pt1;
60                                          it->pt2 = pt2;
61                                          it->rho = rho;
62                                          it->theta = theta;
63                                          it->counter = 0;
64
65                                      }
66                              }
67                              if(insert){
68                                      it++;
69                              }
70                              else{
71                                      break;
72                              }
73                      }
74
75                      if(insert){
76                              // insert line in List
```

```
77                                  lanes.push_back(tmp);
78                                  global_id++;
79                          }
80                          insert = true;
81                  }
82          }
83
84          it2 = lanes.begin();
85          while (it2 != lanes.end()) {
86                  if(!it2->erased){
87                          // draw line in image
88                          cvLine(cropped_color , it2->pt1, it2->pt2, CV_RGB(255, 0, 0), 1, 8 );
89                          it2->counter++;
90                      // lane is shown a little longer if there are discontinuities in the lane
91                          if(it2->counter > 10){
92                                  it2->erased = true;
93                          }
94                  }
95                  it2++;
96          }
97          cvShowImage("linien", cropped_color);
98          cvWaitKey(1);
99
100         cvReleaseImage(&canny);
101         cvReleaseImage(&cropped);
102         cvReleaseImage(&cropped_color);
103 }
```

Im letzten Schritt werden die Fahrspuren in das Videobild eingezeichnet. Um eventuellen Unregelmäßigkeiten oder Unterbrechungen in der Fahrspurmarkierung entgegen zu wirken, werden die Fahrspuren etwas länger eingezeichnet, als sie von unserem Algorithmus gefunden werden. So können eventuelle Lücken ausgeglichen werden.

A.4 Portierung Grafik

Im folgenden wird der Quellcode gelistet, welcher teilweise für die Portierung bearbeitet wurde. Durch den vergleich mit [BBB+09] können so die Änderungen einfach nachfollzogen werden. Die Beschreibung der Module kann in [BBB+09] nachgelesen werden, da die grundlegende Funktionsweise der Module nicht geändert wurde. Falls etwas für die Portierung nicht genutzt wurde, ist dies direkt als Kommentar im Quellcode einzusehen.

Tobias Braun

A.4.1 Arrow: Headerdatei

```
1  ////////////////////////////////////////////////////////////////////////////////
2  //
3  //                      Hochschule Darmstadt - SS09 - Navi3D
4  //                              Copyright by h-da
5  //
6  //                      Created: Marco Muench / 2009/04/21
7  //
8  //                      Person in charge : Marco Muench, Michael Roth
9  //
10 //                      Last Edit by: Marco Muench, 03.8.2009
11 //
12 //                      Create the Arrow for the Navi3D-Application
13 //
14 ////////////////////////////////////////////////////////////////////////////////
15
16 #pragma once
```

```
17
18 #include <iostream>
19 #ifdef WINDOWS
20 #include <windows.h>
21 #endif
22 #include <GL/gl.h>
23 #include "XML/Navi3D_Base.h"
24 #include "ARROW/Navi3D_InterpolatePoints.h"
25 #include <vector>
26 #include <cmath>
27 #include <sstream>
28 #ifndef M_PI
29 #define M_PI    3.14159265358979323846f
30 #endif
31
32 using namespace std;
33
34
35 class Navi3D_Arrow
36 {
37 public:
38         Navi3D_Arrow(void);     //Konstruktor
39         Navi3D_Arrow(Navi3D_Base *tBase, IGADA *tGADA); //Konstruktor
40         ~Navi3D_Arrow(void); //Destruktor
41         void Arrow_Display(); //Display the OpenGL Arrow
42         void changeArrowColor(int newColor); //change Arrow Color with a fixed value
43         void changeArrowSegmentation(); //only in Debug and don?t work with Speed Control /  recolor
               every single sekment
44         void setMaxShadow(float newMaxShadow); //set the maximal Shadow on the Arrow / separate from
               OpenGL Shadow
45         bool setBaseFactor(float factor); //set Base Faktor = size of the Base from arrow Peak
46         bool setNewArrowProfile(int profileNr); // set a new arrow profile from the XML file
47         bool setArrowPeakType(int type); // set new peak type separate from the XML profile
48         bool setArrowBegin(int type); // set arrow begin
49         bool calculateArrowData(); //Calculate the Arrow Data / is needed before you can display the
               arrow
50         char* getNextCurveInMeter(); //get a the next Curve in Meter as char / needed to Display as
               Text on Screen
51
52 private:
53
54         // Variables
55         ////////////
56         Navi3D_Base *pBase;     //Pointer on Base (XML File)
57         N3D_System systemPar;
58         IGADA *pGADA; //Pointer on GADA-Interface
59         N3D_BasePoints arrowHead; //For Arrow Head Navigation Point
60         vector<N3D_BasePoints> vBasePointsArrow; //Base for arrow from the XML file
61         vector<N3D_BasePoints> vBasePointsArrowHead; //Base for arrow peak from the XML file
62         vector<N3D_BasePoints> vCalculateBasePointsArrowHead; //rotate base for arrow peak
63         vector< vector<N3D_BasePoints> > rotBasePoints; //rotate base for arrow
64         int arrowColor; //the current color (0 to 6)
65         bool ArrowSegmentation; //is arrow segmentation aktive (true = on, false = off)
66         float arrowColorR; //amount of Color Red (0 to 1)
67         float arrowColorG; //amount of Color Green (0 to 1)
68         float arrowColorB; //amount of Color Blue (0 to 1)
69         float getPointX(int index); //get X value from Gada-Interface
70         float getPointY(int index); //get Y value from Gada-Interface
71         float getPointZ(int index);     //get Z value from Gada-Interface
72         float powerShadowing[360]; // array of Shadowvalue for each point of base
73         float Shadowing; //maximal Shadowing (0=noShadow to 1=black)
74         bool isCurveOnStreet; //Curve on the street
75         int firstCurveOnStreet; //Navigation point of the first Curve
76         int ArrowPeakType; //current Arrow Peak Type (1 to 7) look createArrowHead(int ArrowNumber,
               N3D_BasePoints &arrowHead)
77         int beginType; //current Arrow begin (0 to 2) look createArrowStart(int variant)
78         float arrowHeadLength; //current Length of the arrow Peak
```

```
79          Colors coloredPartArrow[1000]; //Color for every segment of the arrow /needed for speed
                control /1000 for Beckerdemo
80          int arrowDangerZone[1000]; // array of a Danger Zone for every segment (0 = no Danger, 1= mid.
                Danger, 2 = High Danger)
81          Colors ColorArrow; //mainColor of Arrow
82          Colors ColorDanger; //Color of a danger Arrow  (red)
83          Colors ColorLookOut; //Color of a lookOut Arrow (orange)
84          int arrowProfilNumber; //Profile number for XML File
85          float baseFactor; //current Base size for Peak (baseFactor<1 = smaller /baseFactor>1 = bigger)
86          float maxHightBaseYAxis; //highest Point of Base /needed for the Arrow Head
87          bool increaseBaseCoordinatesForArrowHead; //base stretched (true = Base Stretched , false=
                Base normal) /needed for arrow peak
88          float deepestPointofBase; //deepest Point of Base
89          float pi; //Pi
90          float angleFor2DAArrowPeak; //calculate angle between Vektor(last navigation point to arrow
                peak point) and x-axis
91
92          // Functions
93          ////////////
94          void calculateShadowing(); // calucalte Shadowing  It's separate from OpenGL Shadow
95          void increaseBase(vector<N3D_BasePoints> &vBasePoints, float factor, bool onlyXAxis); //change
                Base with factor a (a<1 -> smaller / a>1 -> bigger)
96          bool calculateColorPartsForArrow(int speed); //calculate the arrow color for every segment
97          double getDistanceBetweenTwoNavPoints(int indexStart, int indexEnd); //get distance between
                two Nav Points
98          void setBasePointsArrow(int profil); //set new vBasePointsArrow by reference to Profile number
99          void setBasePointsArrowHead(int profil); //set new vBasePointsArrowHead by reference to
                Profile number
100         bool createArrowStart(int variant); //create the Arrow Start
101         bool createArrowHead(int ArrowNumber, N3D_BasePoints &arrowHead); //create the Arrow Head
102         void basePointRotate(GAVertex last, GAVertex current, GAVertex next, vector<N3D_BasePoints>&
                points); //rotate Base between three navigation points
103         void setArrowColor(float R, float G, float B); //set new Color RGB
104         //vector helper functions using GAVertex
105         bool equal(GAVertex, GAVertex);  //check two vectors GAVertex if equal
106         GAVertex crossProduct(GAVertex, GAVertex); //compute crossprodukt
107         GAVertex normalize(GAVertex); //normalize a GAVertex
108         GAVertex rotateY(GAVertex, double alpha); //rotate vector GAVertex
109         GAVertex getVector(GAVertex start, GAVertex end); //get Vektor between two GAVertex
110         double getLength(GAVertex); //get Lenght of a vector GAVertex
111         double dotProduct(GAVertex, GAVertex); //compute dotProduct
112         GAVertex add(GAVertex, GAVertex); //add two vectors
113         GAVertex substract(GAVertex, GAVertex); //substract two vectors
114         N3D_BasePoints substract(N3D_BasePoints, GAVertex); //substract a vector from a Navigation
                point
115         GAVertex multiply(GAVertex, double faktor); //multiply two vectors
116         GAVertex divide(GAVertex, double faktor); //divide two vectors
117         double getAngle(GAVertex, GAVertex, GAVertex); //get Angle between two Vectors
118
119 };
```

A.4.2 Arrow: Sourcedatei

```
1  #include "ARROW/Navi3D_Arrow.h"
2  #include <stdio.h>
3  #include <GL/gl.h>
4
5  extern void printLog(string Log, unsigned int Level);
6
7  #define red 1.0,0.0,0.0
8  #define green 0.0,1.0,0.0
9  #define blue 0.0,0.0,1.0
10 #define pink 1.0,0.0,1.0
11 #define yellow 1.0,1.0,0.0
12 #define turquoise 0.0,1.0,1.0
13 #define white 1.0,1.0,1.0
14 #define black 0.0,0.0,0.0
```

```
15
16  bool operator==(Colors a, Colors b)
17  {
18          if((a.r == b.r) && (a.g == b.g) && (a.b == b.b))
19                  return true;
20          else
21                  return false;
22  }
23
24  bool operator!=(Colors a, Colors b)
25  {
26          if((a.r != b.r) || (a.g != b.g) || (a.b != b.b))
27                  return true;
28          else
29                  return false;
30  }
31
32
33
34  Navi3D_Arrow::Navi3D_Arrow(void)
35  {
36
37
38  }
39
40  ////////////////////////////////////////
41  // Constructor
42  ////////////////////////////////////////
43  Navi3D_Arrow::Navi3D_Arrow(Navi3D_Base *tBase, IGADA *tGADA)
44  {
45          pBase = tBase;
46          systemPar = pBase->getSystemParameters();
47          pGADA = tGADA;
48
49          changeArrowColor(2); //setStandardcolor
50          ArrowSegmentation = false;      //Set Debug Segemtation
51          increaseBaseCoordinatesForArrowHead = false;
52
53          Shadowing = 0.6f;
54          arrowHeadLength = 3;
55
56          arrowProfilNumber = 1;
57
58          baseFactor = 1.5f;
59          ArrowPeakType = 5;
60          beginType = 3;
61
62          setNewArrowProfile(arrowProfilNumber);
63          pi = (float) M_PI;
64
65          //Color Danger = Red
66          ColorDanger.r=1.0;
67          ColorDanger.g=0.0;
68          ColorDanger.b=0.0;
69          // Color lookOut = yellow
70          ColorLookOut.r=1.0;
71          ColorLookOut.g=0.5;
72          ColorLookOut.b=0.0;
73
74          printLog("Navi3D_Arrow.cpp: Load constructor successfull ",2);
75  }
76
77  ////////////////////////////////////////
78  // Destructor
79  ////////////////////////////////////////
80  Navi3D_Arrow::~Navi3D_Arrow(void)
81  {
```

```
82          printLog("Navi3D_Arrow.cpp: Destruction successfull ",2);
83  }
84
85  ////////////////////////////////////////
86  // Change the Arror Color with:
87  // int 0 = red
88  // int 1 = green
89  // int 2 = blue
90  // int 3 = pink
91  // int 4 = yellow
92  // int 5 = turquoise
93  // int 6 = white
94  // default: blue
95  ////////////////////////////////////////
96  void Navi3D_Arrow::changeArrowColor(int newColor)
97  {
98          switch(newColor)
99          {
100                 case 0 : arrowColor = 0;
101                                 setArrowColor(red); break;
102                 case 1 : arrowColor = 1;
103                                 setArrowColor(green); break;
104                 case 2 : arrowColor = 2;
105                                 setArrowColor(blue); break;
106                 case 3 : arrowColor = 3;
107                                 setArrowColor(pink); break;
108                 case 4 : arrowColor = 4;
109                                 setArrowColor(yellow); break;
110                 case 5 : arrowColor = 5;
111                                 setArrowColor(turquoise); break;
112                 case 6 : arrowColor = 6;
113                                 setArrowColor(white); break;
114                 default:
115                                 arrowColor = 2;
116                                 setArrowColor(blue);
117
118         }
119         printLog("Navi3D_Arrow.cpp: Change arrow color successfull",3);
120 }
121 ////////////////////////////////////////
122 // set Arrow Color with R=Red, G=Green, B=Blue
123 ////////////////////////////////////////
124 void Navi3D_Arrow::setArrowColor(float R, float G, float B)
125 {
126 if(systemPar.KERNEL.arrowSpeedCollor.enable)
127 {
128         //if Arrow Color is red; the Speed control didnt work
129         if(arrowColor == 0)
130                 changeArrowColor(2); //set the Color on blue
131
132         ColorArrow.r = R;
133         ColorArrow.g = G;
134         ColorArrow.b = B;
135 }
136         arrowColorR = R;
137         arrowColorG = G;
138         arrowColorB = B;
139
140         printLog("Navi3D_Arrow.cpp: Set new arrow color successfull",3);
141 }
142
143 ////////////////////////////////////////
144 // switch the ArrowSegmentation On or Off
145 ////////////////////////////////////////
146 void Navi3D_Arrow::changeArrowSegmentation()
147 {
148         if(ArrowSegmentation)
```

```
149         {
150                 ArrowSegmentation = false;
151                 printLog("Navi3D_Arrow.cpp: Finish arrow segmentation",3);
152         }
153         else
154         {
155                 ArrowSegmentation = true;
156                 printLog("Navi3D_Arrow.cpp: Enable arrow segmentation",3);
157         }
158 }
159
160
161 ////////////////////////////////////////
162 // get the x Axis NavPoint on index
163 ////////////////////////////////////////
164 float Navi3D_Arrow::getPointX(int index)
165 {
166         if(pGADA->verticesCount() > 0)
167                 return (pGADA->getVertex(index).x);
168         return 0;
169 }
170 ////////////////////////////////////////
171 // get the Y Axis NavPoint on index
172 ////////////////////////////////////////
173 float Navi3D_Arrow::getPointY(int index)
174 {
175         if(pGADA->verticesCount() > 0)
176                 return pGADA->getVertex(index).y;
177         return 0;
178 }
179 ////////////////////////////////////////
180 // get the Z Axis NavPoint on index
181 ////////////////////////////////////////
182 float Navi3D_Arrow::getPointZ(int index)
183 {
184         if(pGADA->verticesCount() > 0)
185                 return (pGADA->getVertex(index).z);
186         return 0;
187 }
188
189 ////////////////////////////////////////
190 // load the Base for Arrow
191 ////////////////////////////////////////
192 void Navi3D_Arrow::setBasePointsArrow(int profil)
193 {
194         if(pBase->getBasePoints(&vBasePointsArrow, profil))
195                 calculateShadowing();
196         else
197                 printLog("Navi3D_Arrow.cpp: Error to load the Base for Arrow from Navi3D_Base.cpp",1);
198
199         vBasePointsArrow.push_back(     vBasePointsArrow[0]);
200
201         // find the deepest Point of the Base
202         // Needed for the radiused begin of the Arrow
203         deepestPointofBase = vBasePointsArrow[0].y ;
204
205         for(int i = 1; i < static_cast<int>(vBasePointsArrow.size()); i++) //run all Base-Points-
              Koordinats
206         {
207                 if(vBasePointsArrow[i].y < deepestPointofBase)
208                         deepestPointofBase = vBasePointsArrow[i].y;
209         }
210 #ifdef RADIUSEDSTART
211         N3D_BasePoints newBase;
212         for(float rad = 90; rad < 190; rad+=10)
213         {
```

```
214                         for(int i = 0; i < static_cast<int>(vBasePointsArrow.size()); i++) //run all
                                Base-Points-Koordinats
215                         {
216                                 newBase.x = vBasePointsArrow[i].x;
217                                 newBase.y = (sin(((float)pi/180) * rad)*(vBasePointsArrow[i].y-
                                        deepestPointofBase))+deepestPointofBase;
218                                 newBase.z = -cos(rad*(float)pi/180.)*(vBasePointsArrow[i].y-
                                        deepestPointofBase);
219                                 radiusedBeginBase.push_back(newBase);
220
221                                 //radiusedBeginBase.push_back(( vBasePointsArrow[i].x + getPointX(0),
                                        //set Point of Polygon
222                                 //                              (sin(((float)pi/180) * rad)*(
                                        vBasePointsArrow[i].y-deepestPointofBase))+deepestPointofBase,
223                                 //                              -cos(rad*(float)pi/180.)*(
                                        vBasePointsArrow[i].y-deepestPointofBase));
224                         }
225         }
226 #endif
227 }
228 ////////////////////////////////////////
229 // load the Base for Arrow Head
230 ////////////////////////////////////////
231 void Navi3D_Arrow::setBasePointsArrowHead(int profil)
232 {
233         if(pBase->getBasePoints(&vBasePointsArrowHead, profil))
234         {
235                 increaseBase(vBasePointsArrowHead, baseFactor, increaseBaseCoordinatesForArrowHead);
236
237                 vBasePointsArrowHead.push_back(vBasePointsArrowHead[0]);
238
239                 //search the highest Y-value from the Base
240                 //Needed for the ArrowHead
241                 maxHightBaseYAxis = vBasePointsArrowHead[0].y;
242
243                 for(int i = 1; i < static_cast<int>(vBasePointsArrowHead.size()); i++)
244                 {
245                         if(vBasePointsArrowHead[i].y > maxHightBaseYAxis)
246                                 maxHightBaseYAxis = vBasePointsArrowHead[i].y;
247                 }
248         }
249         else
250                 printLog("Navi3D_Arrow.cpp: Error to load the Base for Arrow Head from Navi3D_Base.cpp
                        ",1);
251 }
252 ////////////////////////////////////////
253 // calculate the Shadowing from the Base
254 ////////////////////////////////////////
255 void Navi3D_Arrow::calculateShadowing()
256 {
257         //Calculate the Shadowing
258         for(unsigned int i = 0; i < vBasePointsArrow.size()-1; i++)
259         {
260                 powerShadowing[i] = atan((vBasePointsArrow[i].y-vBasePointsArrow[i+1].y) / (
                        vBasePointsArrow[i].x-vBasePointsArrow[i+1].x));
261         }
262         int i = (int)vBasePointsArrow.size()-1;
263         powerShadowing[i] = atan((vBasePointsArrow[i].y-vBasePointsArrow[0].y) / (vBasePointsArrow[i].
                x-vBasePointsArrow[0].x));
264
265         //Test the Shadowing
266         for(unsigned int i = 0; i < vBasePointsArrow.size(); i++)
267         {
268                 if(powerShadowing[i] <0)
269                 {
270                         powerShadowing[i] = -powerShadowing[i];
271                 }
```

```
272                 if(powerShadowing[i] >1)
273                 {
274                         powerShadowing[i] = 1;
275                 }
276         }
277 }
278 ////////////////////////////////////////
279 // set a New Maximal Shadow
280 // from 0 = no Shadow
281 // to   1 = max Shadow
282 ////////////////////////////////////////
283 void Navi3D_Arrow::setMaxShadow(float newMaxShadow)
284 {
285         if(newMaxShadow > 1)
286                 Shadowing = 1;
287         else if (newMaxShadow < 0)
288                 Shadowing = 0;
289         else
290                 Shadowing = newMaxShadow;
291 }
292 ////////////////////////////////////////
293 // change the Base for Arrow Head
294 ////////////////////////////////////////
295 void Navi3D_Arrow::increaseBase(vector<N3D_BasePoints> &vBasePoints, float factor, bool onlyXAxis)
296 {
297         for(int i = 0; i < static_cast<int>(vBasePoints.size()); i++)
298         {
299                 vBasePoints[i].x *= factor;
300                 if(!onlyXAxis)vBasePoints[i].y *= factor;
301         }
302 }
303
304 ////////////////////////////////////////
305 // Calculate ArrowData
306 //
307 // Needed before the Arrow can render
308 ////////////////////////////////////////
309 bool Navi3D_Arrow::calculateArrowData()
310 {
311         //recalculate basepoints for rotation
312         rotBasePoints.clear();
313         for(int i = 0; i < pGADA->verticesCount(); i++)
314         {
315                         rotBasePoints.push_back(vBasePointsArrow);
316
317                         if(i > 0 && i < pGADA->verticesCount() -1)
318                                 basePointRotate(pGADA->getVertex(i-1), pGADA->getVertex(i), pGADA->
                                        getVertex(i+1), rotBasePoints[i]);
319                         else if(i > 0)
320                         {
321                                 basePointRotate(pGADA->getVertex(i-1), pGADA->getVertex(i), pGADA->
                                        getVertex(i), rotBasePoints[i]);
322                         }
323         }
324
325
326         //Calculate colors of the Arrow
327         // for Speed Control
328         if(systemPar.KERNEL.arrowSpeedCollor.enable)
329         {
330                 if(systemPar.GADA.mode == BECKER_DEMO || systemPar.GADA.mode == STANDARD_ARROW)
331                 {
332                         calculateColorPartsForArrow(systemPar.KERNEL.arrowSpeedCollor.testSpeed);
333                 }
334                 else
335                 {
336                         calculateColorPartsForArrow(pGADA->getSpeed());
```

```
337                 }
338 }
339
340         //calculate the Arrow Peak Point in x/y/z Koordinates
341         if(systemPar.KERNEL.interpolPoints.enable)
342         {
343                 arrowHead.x = getPointX(pGADA->verticesCount()-1)-getPointX(pGADA->verticesCount()-
                        systemPar.KERNEL.interpolPoints.resolution);
344                 arrowHead.y = getPointY(pGADA->verticesCount()-1)-getPointY(pGADA->verticesCount()-
                        systemPar.KERNEL.interpolPoints.resolution);
345                 arrowHead.z = getPointZ(pGADA->verticesCount()-1)-getPointZ(pGADA->verticesCount()-
                        systemPar.KERNEL.interpolPoints.resolution);
346
347
348                 //norminate ArrowHead Koordinates and calculate ArrowHead Koordinats
349                 float divisor = sqrt((arrowHead.x*arrowHead.x)+(arrowHead.y*arrowHead.y)+(arrowHead.z*
                        arrowHead.z));
350
351                 arrowHead.x = getPointX(pGADA->verticesCount()-1) + (arrowHead.x/divisor)*
                        arrowHeadLength;
352                 arrowHead.y = getPointY(pGADA->verticesCount()-1) + (arrowHead.y/divisor)*
                        arrowHeadLength;
353                 arrowHead.z = getPointZ(pGADA->verticesCount()-1) + (arrowHead.z/divisor)*
                        arrowHeadLength;
354
355         }else{
356
357                 arrowHead.x = getPointX(pGADA->verticesCount()-1)-getPointX(pGADA->verticesCount()-2);
358                 arrowHead.y = getPointY(pGADA->verticesCount()-1)-getPointY(pGADA->verticesCount()-2);
359                 arrowHead.z = getPointZ(pGADA->verticesCount()-1)-getPointZ(pGADA->verticesCount()-2);
360
361
362                 //norminate ArrowHead Koordinates and calculate ArrowHead Koordinats
363                 float divisor = sqrt((arrowHead.x*arrowHead.x)+(arrowHead.y*arrowHead.y)+(arrowHead.z*
                        arrowHead.z));
364
365                 arrowHead.x = getPointX(pGADA->verticesCount()-1) + (arrowHead.x/divisor)*
                        arrowHeadLength;
366                 arrowHead.y = getPointY(pGADA->verticesCount()-1) + (arrowHead.y/divisor)*
                        arrowHeadLength;
367                 arrowHead.z = getPointZ(pGADA->verticesCount()-1) + (arrowHead.z/divisor)*
                        arrowHeadLength;
368 }
369
370         //calculate the Spin of the Arrow Head
371         // needed for Arrow Head Type 1-4,6-7
372         if(ArrowPeakType!=5)
373         {
374                 vCalculateBasePointsArrowHead =  vBasePointsArrowHead;
375                 basePointRotate(pGADA->getVertex(pGADA->verticesCount()-2), pGADA->getVertex(pGADA->
                        verticesCount()-1), pGADA->getVertex(pGADA->verticesCount()-1),
                        vCalculateBasePointsArrowHead);
376         }
377         else
378         {
379                 // Calculate Angle for 2D Arrow
380
381                 GAVertex first; //create GAVertex which is parallel to z-axis
382                 first.x = pGADA->getVertex(pGADA->verticesCount()-1).x;
383                 first.y = pGADA->getVertex(pGADA->verticesCount()-1).y;
384                 first.z = pGADA->getVertex(pGADA->verticesCount()-1).z +arrowHeadLength;
385
386                 GAVertex head;
387                 head.x = arrowHead.x;
388                 head.y = arrowHead.y;
389                 head.z = arrowHead.z;
390
```

```
391                 //calculate the angle between z-axis an the new Arrow Peak
392                 angleFor2DAArrowPeak = (float)getAngle(first ,pGADA->getVertex(pGADA->verticesCount()
                        -1), head );
393                 if(first.x < head.x) //angle is always positiv, but we need sign +/-
394                         angleFor2DAArrowPeak*=-1;
395         }
396
397
398         return true;
399 }
400
401
402 ////////////////////////////////////////
403 // Display the OpenGLArrow
404 ////////////////////////////////////////
405 void Navi3D_Arrow::Arrow_Display()
406 {
407         //pGADA->switchBufferIfNew();
408         if(ArrowSegmentation)
409         {
410                 arrowColor = 0;
411         }
412
413         ////////////////////////////
414         // Start Front Base of Arrow
415 #ifdef RADIUSEDSTART
416         createArrowStart(beginType);
417 #else
418         createArrowStart(1);
419 #endif
420         // END Front Base of Arrow
421         ////////////////////////////
422
423         ////////////////////////////
424         // Start Arrow
425         //vBasePointsArrow.push_back(vBasePointsArrow[0]);
426
427         for(int j=0; j < pGADA->verticesCount()-1; j++) //all NavPoint without Last Point
428         {
429                 //If the Arrow with different Colors
430                 if(ArrowSegmentation)
431                 {
432                         arrowColor++;
433                         changeArrowColor(arrowColor); //change the Color
434                         if(arrowColor > 6)
435                         {
436                                 arrowColor =0;
437                         }
438                 }
439
440                 //create a Arrow Part
441                 glBegin( GL_QUADS );
442
443                         for(int i = 0; i < static_cast<int>(vBasePointsArrow.size()-1); i++)
444                         {
445
446 if(systemPar.KERNEL.arrowSpeedCollor.enable)
447 {
448                                 glColor3f(  coloredPartArrow[j+1].r - Shadowing*powerShadowing[i],
449                                             coloredPartArrow[j+1].g - Shadowing*
                                                    powerShadowing[i],
450                                             coloredPartArrow[j+1].b - Shadowing*
                                                    powerShadowing[i]);
451 }else{
452                                 glColor3f(  arrowColorR - Shadowing*powerShadowing[i],
453                                             arrowColorG - Shadowing*powerShadowing[i],
454                                             arrowColorB - Shadowing*powerShadowing[i]);
```

```
455 }
456
457                                 glVertex3f( rotBasePoints[j+1][i].x + getPointX(j+1),
458                                             rotBasePoints[j+1][i].y + getPointY(j+1),
459                                             rotBasePoints[j+1][i].z + getPointZ(j+1));
460                                 glVertex3f( rotBasePoints[j+1][i+1].x + getPointX(j+1),
461                                             rotBasePoints[j+1][i+1].y + getPointY(j+1),
462                                             rotBasePoints[j+1][i+1].z + getPointZ(j+1));
463
464
465
466                                 if(systemPar.KERNEL.arrowSpeedCollor.enable)
467                                 {
468                                 glColor3f(  coloredPartArrow[j].r - Shadowing*powerShadowing[i],
469                                             coloredPartArrow[j].g - Shadowing*
                                                 powerShadowing[i],
470                                             coloredPartArrow[j].b - Shadowing*
                                                 powerShadowing[i]);
471                                 }
472
473                                 glVertex3f( rotBasePoints[j][i+1].x + getPointX(j),
474                                             rotBasePoints[j][i+1].y + getPointY(j),
475                                             rotBasePoints[j][i+1].z + getPointZ(j));
476                                 glVertex3f( rotBasePoints[j][i].x + getPointX(j),
477                                             rotBasePoints[j][i].y + getPointY(j),
478                                             rotBasePoints[j][i].z + getPointZ(j));
479                         }
480
481                 glEnd();
482         }
483         // END Arrow
484         ////////////////////////////
485
486         //Create the Arrow Peak
487         createArrowHead(ArrowPeakType, arrowHead);
488 }
489
490
491 ////////////////////////////////////////
492 // spin the Base
493 ////////////////////////////////////////
494 void Navi3D_Arrow::basePointRotate(GAVertex A, GAVertex B, GAVertex C, vector<N3D_BasePoints>& points)
495 {
496
497 #ifndef ROTATE
498         //If rotate is deactivated by define, do nothing
499         return;
500 #endif
501         //Input: points: the arrows basepoints in x/y plane
502         //Input: A,B,C: navigational points in x/z plane
503         //Note: This function calculates the rotated base points belonging to point B
504
505         //This function utilizes two-dimensional homogenous coordinates, so everything has
506         //      to be converted to x/y plane with z as the homogenous coordinate w
507         //Also here all calculations are done in 2D, to make it faster.
508
509         vector<N3D_BasePoints> tmp = points;
510
511         //basepoints are projected to x/z plane and z and y are switched, so basically
512         //      only the x coordinates are used.
513         //Note: w has to be 1 by definition of homogenous coordinated
514         for(unsigned int i = 0; i < tmp.size(); i++)
515         {
516                 tmp[i].y = 0;
517                 tmp[i].z = 1;
518         }
519
```

```
520
521         //Same as above for the navigational points
522         A.y = A.z;
523         B.y = B.z;
524         C.y = C.z;
525
526         A.z = 1;
527         B.z = 1;
528         C.z = 1;
529
530         GAVertex vAB, vBC;
531
532         //the two main vectors are calculated
533         vAB = getVector(A,B);
534         vBC = getVector(B,C);
535
536         //the normal of a vector is the cross product of end and start point
537         GAVertex vABnormal = crossProduct(B,A);
538         GAVertex vBCnormal = crossProduct(C,B);
539
540         //a directional vectors w ( or z ) coordinate is 0
541         GAVertex vABdirNormal = vABnormal;
542         GAVertex vBCdirNormal = vBCnormal;
543         vABdirNormal.z = 0;
544         vBCdirNormal.z = 0;
545
546         //a normalized normal still points to the same direction, but has the length 1
547         GAVertex vABnNormal = normalize(vABdirNormal);
548         GAVertex vBCnNormal = normalize(vBCdirNormal);
549
550         GAVertex pABs, pABe, pBCs, pBCe, schnitt, pABn, pBCn;
551         for(unsigned int i = 0; i < tmp.size(); i++)
552         {
553                 //this constructs start and end points of a line parallel to vAB and vBC,
554                 //      which is done by multiplying the normalized normal with the X value of
555                 //      a basepoint and then adding it to A,B or accordingly
556                 pABs = add(A, multiply(vABnNormal, -tmp[i].x));
557                 pABe = add(B, multiply(vABnNormal, -tmp[i].x));
558
559                 pBCs = add(B, multiply(vBCnNormal, -tmp[i].x));
560                 pBCe = add(C, multiply(vBCnNormal, -tmp[i].x));
561
562                 //then the normals of those parallels are computed
563                 pABn = crossProduct(pABe, pABs);
564                 pBCn = crossProduct(pBCe, pBCs);
565
566                 //the intersection of two lines is calculated by the cross product of the two normals
567                 schnitt = crossProduct(pBCn, pABn);
568
569
570                 //if w is 0 (or close to 0 considering rounding errors), pAB and pBC are parallel
571                 //if they are not parallel, the intersection has to be divided by its w component
572                 if(abs(schnitt.z) > 0.01f && equal(B,C) == false)
573                         schnitt = divide(schnitt, schnitt.z);
574                 else
575                 {
576                         //if the lines were parallel, the intersection is set to pAB endpoint
577                         schnitt = pABe;
578                 }
579
580                 //translate the intersection to point B
581                 schnitt = substract(schnitt, B);
582
583                 //set x and z coordinates, y stays untouched
584                 points[i].x = schnitt.x;
585                 points[i].z = schnitt.y;
586
```

```
587         }
588
589 }
590
591 ////////////////////////////////////////
592 // vector helper functions
593 ////////////////////////////////////////
594
595 //Simple equality function
596 bool Navi3D_Arrow::equal(GAVertex a, GAVertex b)
597 {
598         if(a.x != b.x)
599                 return false;
600         if(a.y != b.y)
601                 return false;
602         if(a.z != b.z)
603                 return false;
604
605         return true;
606 }
607
608 //Crossproduct for two three-dimensional vectors
609 GAVertex Navi3D_Arrow::crossProduct(GAVertex a, GAVertex b)
610 {
611         GAVertex tmp;
612         tmp.x = a.y * b.z - a.z * b.y;
613         tmp.y = a.z * b.x - a.x * b.z;
614         tmp.z = a.x * b.y - a.y * b.x;
615
616         return tmp;
617 }
618
619 //Multiply a vector with a scalar factor
620 GAVertex Navi3D_Arrow::multiply(GAVertex v, double faktor)
621 {
622         GAVertex tmp;
623         tmp.x = (float)(v.x * faktor);
624         tmp.y = (float)(v.y * faktor);
625         tmp.z = (float)(v.z * faktor);
626
627         return tmp;
628 }
629
630 //Dividy a vector by a scalar
631 GAVertex Navi3D_Arrow::divide(GAVertex v, double faktor)
632 {
633         GAVertex tmp;
634         tmp.x = (float)( v.x / faktor);
635         tmp.y = (float)( v.y / faktor);
636         tmp.z = (float)( v.z / faktor);
637
638         return tmp;
639 }
640
641 //Normalize a vector by dividing it by its length
642 GAVertex Navi3D_Arrow::normalize(GAVertex v)
643 {
644         GAVertex tmp;
645         double length = getLength(v);
646         tmp.x = (float) (v.x / length);
647         tmp.y = (float) (v.y / length);
648         tmp.z = (float) (v.z / length);
649
650         return tmp;
651 }
652
653 //Rotate a point arround Y Axis
```

```
654 GAVertex Navi3D_Arrow::rotateY(GAVertex v, double alpha)
655 {
656         GAVertex tmp;
657         float x = v.x;
658         float y = v.y;
659         float z = v.z;
660 
661         tmp.x = (x * cos((float)alpha)) +  (z * -sin((float)alpha));
662         tmp.y = y;
663         tmp.z = (x * sin((float)alpha)) + (z * cos((float)alpha));
664 
665         return tmp;
666 }
667 
668 //Get directional vector between two points
669 GAVertex Navi3D_Arrow::getVector(GAVertex start, GAVertex end)
670 {
671         GAVertex erg;
672         erg.x = end.x - start.x;
673         erg.y = end.y - start.y;
674         erg.z = end.z - start.z;
675 
676         return erg;
677 }
678 
679 //Get the lengts of a directional vector
680 double Navi3D_Arrow::getLength(GAVertex v)
681 {
682         return(sqrt(v.x * v.x + v.y * v.y + v.z * v.z));
683 }
684 
685 //Get the dot product of two vectors
686 double Navi3D_Arrow::dotProduct(GAVertex v1, GAVertex v2)
687 {
688         return (v1.x * v2.x + v1.y * v2.y + v1.z * v2.z);
689 }
690 
691 //Add two vectors
692 GAVertex Navi3D_Arrow::add(GAVertex v1, GAVertex v2)
693 {
694         GAVertex tmp;
695         tmp.x = v1.x + v2.x;
696         tmp.y = v1.y + v2.y;
697         tmp.z = v1.z + v2.z;
698 
699         return tmp;
700 }
701 
702 //Substract two vectors
703 GAVertex Navi3D_Arrow::substract(GAVertex v1, GAVertex v2)
704 {
705         GAVertex tmp;
706         tmp.x = v1.x - v2.x;
707         tmp.y = v1.y - v2.y;
708         tmp.z = v1.z - v2.z;
709 
710         return tmp;
711 }
712 
713 //Another substract function, for convinience only
714 N3D_BasePoints Navi3D_Arrow::substract(N3D_BasePoints v1, GAVertex v2)
715 {
716         N3D_BasePoints tmp;
717         tmp.x = v1.x - v2.x;
718         tmp.y = v1.y - v2.y;
719         tmp.z = v1.z - v2.z;
720 
```

```
721         return tmp;
722 }
723
724 //Get the angle between two directional vectors in degrees
725 //Note: a, b and c are points
726 //Note: vector 1 spans between point a and b, vector 2 between b and c
727 //Note: the function returns always the angle <180
728 double Navi3D_Arrow::getAngle(GAVertex a, GAVertex b, GAVertex c)
729 {
730         double alpha = acos(dotProduct( normalize(getVector(a,b)) , normalize(getVector(b,c)) ));
731
732         alpha = (alpha / M_PI) * 180;
733
734         return 180-alpha;
735 }
736
737 ////////////////////////////////////////
738 // set a new Base Factor
739 ////////////////////////////////////////
740 bool Navi3D_Arrow::setBaseFactor(float factor)
741 {
742         baseFactor = factor;
743         setBasePointsArrowHead(arrowProfilNumber);
744         return true;
745 }
746
747 ////////////////////////////////////////
748 // set a new Arrow Typ with Profile Number
749 ////////////////////////////////////////
750 bool Navi3D_Arrow::setNewArrowProfile(int profileNr)
751 {
752         arrowProfilNumber = profileNr;
753         setBasePointsArrow(arrowProfilNumber);
754         setBasePointsArrowHead(arrowProfilNumber);
755         return true;
756 }
757
758 ////////////////////////////////////////
759 // Display the Arrow Head
760 //
761 // ArrowNumber = 1 : Base stretch in X and Y Axis, ArrowPeak in the Middle
762 // ArrowNumber = 2 : Base stretch only in X Axis, ArrowPeak in the Middle
763 // ArrowNumber = 3 : Base stretch in X and Y Axis, ArrowPeak above the Middle
764 // ArrowNumber = 4 : Base stretch only in X Axis, ArrowPeak above the Middle
765 // ArrowNumber = 5 : 2D Arrow
766 // ArrowNumber = 6 : Better Arrow with soft passage, ArrowPeak in the Middle
767 // Arrownumber = 7 : Better Arrow with soft passage, ArrowPeak above the Middle
768 ////////////////////////////////////////
769 bool Navi3D_Arrow::createArrowHead(int ArrowNumber, N3D_BasePoints &arrowHead)
770 {
771         switch(ArrowNumber)
772         {
773         case 1:
774                 if(increaseBaseCoordinatesForArrowHead) //change the
775                 {
776                         increaseBaseCoordinatesForArrowHead = false;
777                         setBasePointsArrowHead(arrowProfilNumber);
778                         maxHightBaseYAxis = 0;
779                 }
780         case 2:
781                 if(!increaseBaseCoordinatesForArrowHead && ArrowNumber == 2)
782                 {
783                         increaseBaseCoordinatesForArrowHead = true;
784                         setBasePointsArrowHead(arrowProfilNumber);
785                         maxHightBaseYAxis = 0;
786                 }
787         case 3:
```

```
788                 if(increaseBaseCoordinatesForArrowHead && ArrowNumber == 3)
789                 {
790                         increaseBaseCoordinatesForArrowHead = false;
791                         setBasePointsArrowHead(arrowProfilNumber);
792                 }
793         case 4:
794                 {
795                 if(!increaseBaseCoordinatesForArrowHead && ArrowNumber == 4)
796                 {
797                         increaseBaseCoordinatesForArrowHead = true;
798                         setBasePointsArrowHead(arrowProfilNumber);
799                 }
800
801                 ////Behind Arrow Head
802                         glBegin( GL_POLYGON);    //create Polygon
803                                 glColor3f(      arrowColorR - Shadowing,
804                                                 arrowColorG - Shadowing,
805                                                 arrowColorB - Shadowing);
806                                 for(int i = 0; i < static_cast<int>(vBasePointsArrowHead.size()); i++)
                                        //run all Base-Points-Koordinats
807                                 {
808                                         glVertex3f( vCalculateBasePointsArrowHead[i].x +
                                            getPointX(pGADA->verticesCount()-1), //set Point
                                            of Polygon
809                                                                         vCalculateBasePointsArrowHead[
                                                                            i].y + getPointY(pGADA->
                                                                            verticesCount()-1),
810                                                                         vCalculateBasePointsArrowHead[
                                                                            i].z + getPointZ(pGADA->
                                                                            verticesCount()-1));
811                                 }
812                         glEnd();
813
814                         // Arrow Head
815                         for(int i = (int)vBasePointsArrowHead.size()-2; i >=0; i--)
816                         {
817                                 glBegin( GL_TRIANGLES );
818
819                                 glColor3f(  arrowColorR - Shadowing*powerShadowing[i],
820                                                 arrowColorG - Shadowing*powerShadowing[i],
821                                                 arrowColorB - Shadowing*powerShadowing[i]);
822
823                                 glVertex3f( vCalculateBasePointsArrowHead[i].x + getPointX(pGADA->
                                            verticesCount()-1),
824                                                 vCalculateBasePointsArrowHead[i].y + getPointY
                                                    (pGADA->verticesCount()-1),
825                                                 vCalculateBasePointsArrowHead[i].z + getPointZ
                                                    (pGADA->verticesCount()-1));
826                                 glVertex3f( arrowHead.x,
827                                                 arrowHead.y+maxHightBaseYAxis,
828                                                 arrowHead.z);
829                                 glVertex3f( vCalculateBasePointsArrowHead[i+1].x + getPointX(pGADA->
                                            verticesCount()-1),
830                                                 vCalculateBasePointsArrowHead[i+1].y +
                                                    getPointY(pGADA->verticesCount()-1),
831                                                 vCalculateBasePointsArrowHead[i+1].z +
                                                    getPointZ(pGADA->verticesCount()-1));
832                                 glEnd();
833                         }
834
835                                         //Side from Last Point to first Point
836
837                         //vBasePointsArrowHead = tmp;
838                 break;
839                 }
840         case 5:
841                 {
```

```
842                     if(!increaseBaseCoordinatesForArrowHead)
843                     {
844                             increaseBaseCoordinatesForArrowHead = true;
845                             setBasePointsArrowHead(arrowProfilNumber);
846                     }
847
848
849
850                     glBegin( GL_TRIANGLES); //create Polygon
851                             glColor3f(      arrowColorR,
852                                             arrowColorG,
853                                             arrowColorB);
854
855
856                             glVertex3f( arrowHead.x, //Head Vertex
857                                             arrowHead.y+maxHightBaseYAxis,
858                                             arrowHead.z);
859
860                                     glVertex3f( cos(((float)pi/180) * (
                                            angleFor2DAArrowPeak))*2 +pGADA->getVertex(pGADA
                                            ->verticesCount()-1).x,
861                                                     arrowHead.y+maxHightBaseYAxis,
862                                                     sin(((float)pi/180) * (
                                                        angleFor2DAArrowPeak))*2
                                                        +pGADA->getVertex(pGADA->
                                                        verticesCount()-1).z);
863                                     glVertex3f( cos(((float)pi/180) * (
                                            angleFor2DAArrowPeak+180))*2 +pGADA->getVertex(
                                            pGADA->verticesCount()-1).x,
864                                                     arrowHead.y+maxHightBaseYAxis,
865                                                     sin(((float)pi/180) * (
                                                        angleFor2DAArrowPeak+180)
                                                        )*2 +pGADA->getVertex(
                                                        pGADA->verticesCount()-1)
                                                        .z);
866
867                     glEnd();
868             break;
869             }
870     case 6:
871             if(increaseBaseCoordinatesForArrowHead && ArrowNumber == 6)
872             {
873                     increaseBaseCoordinatesForArrowHead = false;
874                     setBasePointsArrowHead(arrowProfilNumber);
875             }
876     case 7:
877             if(!increaseBaseCoordinatesForArrowHead && ArrowNumber == 7)
878             {
879                     increaseBaseCoordinatesForArrowHead = true;
880                     setBasePointsArrowHead(arrowProfilNumber);
881                     maxHightBaseYAxis = 0;
882             }
883             ////////////////////////////////
884             // Arrow to Arrow Peak
885             ////////////////////////////////
886             {
887
888             //vBasePointsArrowHead.push_back(vBasePointsArrowHead[0]);
889
890             //vector<N3D_BasePoints> tmp = vBasePointsArrowHead;
891
892             //basePointRotate(pGADA->verticesCount()-2, pGADA->verticesCount()-1, pGADA->
                    verticesCount()-1 )
893             //basePointRotate(pGADA->getVertex(pGADA->verticesCount()-2), pGADA->getVertex(pGADA->
                    verticesCount()-1), pGADA->getVertex(pGADA->verticesCount()-1),
                    vBasePointsArrowHead);
894
```

```
895
896             for(int i = (int)vBasePointsArrowHead.size()-2; i >=0; i--)
897             {
898                     glBegin( GL_TRIANGLES );
899
900                     glColor3f(  arrowColorR - Shadowing*powerShadowing[i],
901                                 arrowColorG - Shadowing*powerShadowing[i],
902                                 arrowColorB - Shadowing*powerShadowing[i]);
903
904                     glVertex3f( vCalculateBasePointsArrowHead[i].x + getPointX(pGADA->
                            verticesCount()-1),
905                                 vCalculateBasePointsArrowHead[i].y + getPointY(pGADA->
                                    verticesCount()-1),
906                                 vCalculateBasePointsArrowHead[i].z + getPointZ(pGADA->
                                    verticesCount()-1));
907                     glVertex3f( arrowHead.x,
908                                 arrowHead.y+maxHightBaseYAxis,
909                                 arrowHead.z);
910                     glVertex3f( vCalculateBasePointsArrowHead[i+1].x + getPointX(pGADA->
                            verticesCount()-1),
911                                 vCalculateBasePointsArrowHead[i+1].y + getPointY(pGADA
                                    ->verticesCount()-1),
912                                 vCalculateBasePointsArrowHead[i+1].z + getPointZ(pGADA
                                    ->verticesCount()-1));
913                     glEnd();
914             }
915
916                                 //Side from Last Point to first Point
917             ////////////////////////////////
918             // turn Arrow Head to behind
919             ////////////////////////////////
920             for(int i = (int)vBasePointsArrowHead.size()-2; i >=0; i--)
921             {
922                     glBegin( GL_TRIANGLES );
923
924                     glColor3f(  arrowColorR - Shadowing*powerShadowing[i],
925                                 arrowColorG - Shadowing*powerShadowing[i],
926                                 arrowColorB - Shadowing*powerShadowing[i]);
927
928                     glVertex3f( vCalculateBasePointsArrowHead[i].x + getPointX(pGADA->
                            verticesCount()-1),
929                                 vCalculateBasePointsArrowHead[i].y + getPointY(pGADA->
                                    verticesCount()-1),
930                                 vCalculateBasePointsArrowHead[i].z + getPointZ(pGADA->
                                    verticesCount()-1));
931                     glVertex3f( getPointX(pGADA->verticesCount()-1) - (arrowHead.x - getPointX(
                            pGADA->verticesCount()-1))/2,
932                                 getPointY(pGADA->verticesCount()-1) - (arrowHead.y -
                                    getPointY(pGADA->verticesCount()-1)),
933                                 getPointZ(pGADA->verticesCount()-1) - (arrowHead.z -
                                    getPointZ(pGADA->verticesCount()-1))/2);
934                     glVertex3f( vCalculateBasePointsArrowHead[i+1].x + getPointX(pGADA->
                            verticesCount()-1),
935                                 vCalculateBasePointsArrowHead[i+1].y + getPointY(pGADA
                                    ->verticesCount()-1),
936                                 vCalculateBasePointsArrowHead[i+1].z + getPointZ(pGADA
                                    ->verticesCount()-1));
937                     glEnd();
938             }
939
940
941             //vBasePointsArrowHead = tmp;
942             break;
943             }
944
945         default:
946             return false;
```

```
947
948
949         }
950         return true;
951
952 }
953
954
955 ////////////////////////////////////////
956 // Calculate the Color of the Arrow conditioned from speed
957 ////////////////////////////////////////
958 bool Navi3D_Arrow::calculateColorPartsForArrow(int speed)
959 {
960         //if only two Points = no Angle
961 if(systemPar.KERNEL.interpolPoints.enable)
962 {
963         if(pGADA->verticesCount() < (2*systemPar.KERNEL.interpolPoints.resolution))
964         {
965                 for(int i = 0; i< pGADA->verticesCount()-1; i++) //go through the arrow
966                 {
967                         coloredPartArrow[i]= ColorArrow;
968                 }
969                 return false;
970         }
971 }else{
972         if(pGADA->verticesCount() < 3)
973         {
974                 for(int i = 0; i< pGADA->verticesCount()-1; i++) //go through the arrow
975                 {
976                         coloredPartArrow[i]= ColorArrow;
977                 }
978                 return false;
979         }
980 }
981
982   isCurveOnStreet = false;
983   firstCurveOnStreet = 0;
984         //calculate Dangerzones
985         for(int i = 0; i< pGADA->verticesCount()-1; i++)
986         {
987                 //if Start or End = NoDangerZone
988                 //we need three Points to calculate a DangerZone
989                 if(systemPar.KERNEL.interpolPoints.enable)
990                 {
991                         if(i == 0 || i == (pGADA->verticesCount()) || (i%systemPar.KERNEL.
                                interpolPoints.resolution)!=0)
992                         {
993                                                 arrowDangerZone[i]= 0;
994                                                 continue;
995                                         }
996                 }else{
997                         if(i == 0 || i == (pGADA->verticesCount()-1))
998                         {
999                                                 arrowDangerZone[i]= 0;
1000                                                continue;
1001                                        }
1002 }
1003
1004
1005                //// First Step
1006                //calculate angle between three NavPoints
1007                float angle = 0;
1008                if(systemPar.KERNEL.interpolPoints.enable)
1009                {
1010                        angle= (float)getAngle(pGADA->getVertex(i-systemPar.KERNEL.interpolPoints.
                               resolution),
1011                                                pGADA->getVertex(i),
```

```
                                        pGADA->getVertex(i+systemPar.KERNEL.interpolPoints.resolution
                                            -1));
                }
                else{
                        angle = (float) getAngle(pGADA->getVertex(i-1),pGADA->getVertex(i),pGADA->
                            getVertex(i+1));
                }

                if(angle < 45  && speed >30)
                {
                        arrowDangerZone[i] = 2;
                        isCurveOnStreet = true;
                        if(firstCurveOnStreet==0)
                                firstCurveOnStreet=i;
                }
                else if(angle < 45 && speed >20)
                {
                        arrowDangerZone[i] = 1;
                        isCurveOnStreet = true;
                        if(firstCurveOnStreet==0)
                                firstCurveOnStreet=i;
                }
                else if( angle < 100 && speed > 60)
                {
                        arrowDangerZone[i] = 2;
                        isCurveOnStreet = true;
                        if(firstCurveOnStreet==0)
                                firstCurveOnStreet=i;
                }
                else if( angle < 100 && speed > 40)
                {
                        arrowDangerZone[i] = 1;
                        isCurveOnStreet = true;
                        if(firstCurveOnStreet==0)
                                firstCurveOnStreet=i;
                }
                else if (angle < 150 && speed > 130)
                {
                        arrowDangerZone[i] = 2;
                        isCurveOnStreet = true;
                        if(firstCurveOnStreet==0)
                                firstCurveOnStreet=i;
                }
                else if (angle < 150 && speed > 100)
                {
                        arrowDangerZone[i] = 1;
                        isCurveOnStreet = true;
                        if(firstCurveOnStreet==0)
                                firstCurveOnStreet=i;
                }
                else
                {
                        arrowDangerZone[i] = 0;
                }
        }


                //// Second Step
                // Colored the Arrow


                if(!isCurveOnStreet) //no Curve on Street
                {
                        for(int i = 0; i< pGADA->verticesCount()-1; i++)
                        {
                                coloredPartArrow[i] = ColorArrow; //colored all Part with Arrow Color
                        }
```

```
1077                    return true;
1078            }
1079
1080            for(int i = 0; i< pGADA->verticesCount(); i++)
1081            {
1082
1083                    if(arrowDangerZone[i]==2)// if Dangerzone
1084                    {
1085                            //int halfBrakingDistance = speed/2;
1086
1087                            int dangerPoints = speed;
1088
1089                            if(i-dangerPoints<0)
1090                                    dangerPoints = i;
1091
1092                            for(int j = 0; j < dangerPoints; j++)
1093                                    coloredPartArrow[i-j] = ColorDanger; //set Dangerzone
                                            Color
1094
1095                            ////////////////////////
1096                            // Needs Very, Very good CPU
1097                            ////////////////////////
1098                            //for(int j = 0; j <= i; j++)
1099                            //{
1100                            //      if(speed>getDistanceBetweenTwoNavPoints(j,i))
1101                            //              coloredPartArrow[j] = ColorDanger; //set Dangerzone
                                        Color
1102                            //      else
1103                            //              coloredPartArrow[j] = ColorArrow;
1104                            //}
1105                    }
1106                    else if(arrowDangerZone[i]==1)// if Lookout Zone
1107                    {
1108
1109                            ////////////////////////
1110                            // Needs Very, Very good CPU
1111                            ////////////////////////
1112                            //for(int j = 0; j < i; j++)
1113                            //{
1114                            //      if(speed>getDistanceBetweenTwoNavPoints(j,i))
1115                            //      {
1116                            //              if(!(coloredPartArrow[i-j]==ColorDanger))
1117                            //                      coloredPartArrow[j] = ColorLookOut; //set
                                        Dangerzone Color
1118                            //              else
1119                            //                      coloredPartArrow[j] = ColorArrow;
1120                            //      }
1121                            //      else
1122                            //              coloredPartArrow[j] = ColorArrow;
1123                            //}
1124                            int dangerPoints = speed;
1125                            if(i-dangerPoints<0)
1126                                    dangerPoints = i;
1127
1128                            for(int j = 0; j < dangerPoints; j++)
1129                                    if(!(coloredPartArrow[i-j]==ColorDanger)) //if at this Point a
                                            Danger Zone
1130                                      coloredPartArrow[i-j] = ColorLookOut; //set Dangerzone
                                              Color
1131                    }
1132                    else
1133                            coloredPartArrow[i]= ColorArrow;
1134            }
1135
1136        return true;
1137    }
1138
```

```
1139 ////////////////////////////////////////
1140 // create the Arrow Start
1141 //
1142 // ArrowNumber = 1 : normal Arrow Start
1143 // ArrowNumber = 2 : Radiused Start from bottom
1144 // ArrowNumber = 3 : Radiused Start from middle
1145 ////////////////////////////////////////
1146 bool Navi3D_Arrow::createArrowStart(int variant)
1147 {
1148         switch(variant)
1149         {
1150         case 1:
1151                 {
1152                         glBegin( GL_POLYGON);    //create Polygon
1153                                 glColor3f(       arrowColorR - Shadowing,
1154                                                  arrowColorG - Shadowing,
1155                                                  arrowColorB - Shadowing);
1156                                 for(int i = 0; i < static_cast<int>(vBasePointsArrow.size()); i++) //
                                        run all Base-Points-Koordinats
1157                                 {
1158                                         glVertex3f( vBasePointsArrow[i].x + getPointX(0), //
                                                set Point of Polygon
1159                                                                 vBasePointsArrow[i].y +
                                                                        getPointY(0),
1160                                                                 vBasePointsArrow[i].z +
                                                                        getPointZ(0));
1161                                 }
1162                         glEnd();
1163
1164
1165                         ////////////////////////////////////////////////////////////////////
1166                         // used case 1 for concave Bases
1167                         // need the middle in 0/0/0
1168                         ////////////////////////////////////////////////////////////////////
1169                         /*
1170                         glBegin( GL_TRIANGLE_FAN);       //create Polygon
1171                                 glColor3f(       arrowColorR - Shadowing,
1172                                                  arrowColorG - Shadowing,
1173                                                  arrowColorB - Shadowing);
1174
1175                                 glVertex3f( getPointX(0), //set Point of Polygon
1176                                                  getPointY(0),
1177                                                  getPointZ(0));
1178
1179
1180                                 for(int i = 0; i < static_cast<int>(vBasePointsArrow.size()); i++) //
                                        run all Base-Points-Koordinats
1181                                 {
1182                                         glVertex3f( vBasePointsArrow[i].x + getPointX(0), //
                                                set Point of Polygon
1183                                                                 vBasePointsArrow[i].y +
                                                                        getPointY(0),
1184                                                                 vBasePointsArrow[i].z +
                                                                        getPointZ(0));
1185                                 }
1186                         glEnd();
1187                         */
1188                         break;
1189                 }
1190         case 2:
1191                 {
1192                         vector<N3D_BasePoints> thisBase = vBasePointsArrow;
1193                         vector<N3D_BasePoints> nextBase = vBasePointsArrow;
1194
1195                         for(int i = 0; i < static_cast<int>(vBasePointsArrow.size()); i++) //run all
                                Base-Points-Koordinats
1196                                 {
```

```
1197                                    nextBase[i].x += getPointX(0); //set Point of Polygon
1198                                    nextBase[i].y += getPointY(0);
1199                            }
1200
1201                    for(float rad = 85; rad < 190; rad+=10)
1202                    {
1203                            thisBase = nextBase;
1204                            for(int i = 0; i < static_cast<int>(vBasePointsArrow.size()); i++) //
                                    run all Base-Points-Koordinats
1205                            {
1206                                    nextBase[i].x = vBasePointsArrow[i].x + getPointX(0); //set
                                            Point of Polygon
1207                                    nextBase[i].y = (sin(((float)pi/180) * rad)*(vBasePointsArrow[
                                            i].y-deepestPointofBase))+deepestPointofBase;
1208                                    nextBase[i].z = (float)(-cos(rad*(float)pi/180.)*(
                                            vBasePointsArrow[i].y-deepestPointofBase));
1209                            }
1210
1211                            for(int i = 0; i < static_cast<int>(vBasePointsArrow.size()-1); i++)
                                    //run all Base-Points-Koordinats
1212                            {
1213                                    glBegin( GL_QUADS);      //create Quads
1214                                            glColor3f(      arrowColorR - Shadowing*powerShadowing
                                                    [i],
1215                                                                    arrowColorG - Shadowing*
                                                                            powerShadowing[i],
1216                                                                    arrowColorB - Shadowing*
                                                                            powerShadowing[i]);
1217                                            glVertex3f( thisBase[i].x , thisBase[i].y + getPointY
                                                    (0), thisBase[i].z + getPointZ(0));
1218                                            glVertex3f( nextBase[i].x , nextBase[i].y + getPointY
                                                    (0), nextBase[i].z + getPointZ(0));
1219                                            glVertex3f( nextBase[i+1].x , nextBase[i+1].y +
                                                    getPointY(0), nextBase[i+1].z + getPointZ(0));
1220                                            glVertex3f( thisBase[i+1].x , thisBase[i+1].y +
                                                    getPointY(0), thisBase[i+1].z + getPointZ(0));
1221                                    glEnd();
1222                            }
1223                    }
1224                    break;
1225            }
1226    case 3:
1227            {
1228                    vector <N3D_BasePoints> nextBase = vBasePointsArrow;
1229                    vector <N3D_BasePoints> lastBase = vBasePointsArrow;
1230                    float cosinus, sinus;
1231
1232                    for(int i = 0; i<=90; i+=5)
1233                    {
1234                            lastBase = nextBase;
1235                            cosinus = (float)(cos(i*(float)pi/180.));
1236                            sinus = sin(i*(float)pi/180);
1237                            if(sinus > 0.785)
1238                                    sinus = 0.785f;
1239
1240
1241                            for(int j = 0; j < static_cast<int>(vBasePointsArrow.size()); j++)
1242                            {
1243                                    nextBase[j].x *=cosinus;
1244                                    nextBase[j].y *=cosinus;
1245                                    nextBase[j].z = sinus;
1246                            }
1247
1248                            for(int j = 0; j < static_cast<int>(vBasePointsArrow.size()-1); j++)
                                    //run all Base-Points-Koordinats
1249                            {
1250                                    glBegin( GL_POLYGON);   //create Polygon
```

```
1251                                    glColor3f(      arrowColorR - Shadowing*powerShadowing[j],
1252                                                    arrowColorG - Shadowing*powerShadowing
                                                        [j],
1253                                                    arrowColorB - Shadowing*powerShadowing
                                                        [j]);
1254
1255                                    glVertex3f( lastBase[j].x+ getPointX(0), lastBase[j].y+
                                            getPointY(0), lastBase[j].z+ getPointZ(0));
1256                                    glVertex3f( nextBase[j].x+ getPointX(0), nextBase[j].y+
                                            getPointY(0), nextBase[j].z+ getPointZ(0));
1257                                    glVertex3f( nextBase[j+1].x+ getPointX(0), nextBase[j+1].y+
                                            getPointY(0), nextBase[j+1].z+ getPointZ(0));
1258                                    glVertex3f( lastBase[j+1].x+ getPointX(0), lastBase[j+1].y+
                                            getPointY(0), lastBase[j+1].z+ getPointZ(0));
1259
1260                                    glEnd();
1261                            }
1262
1263                    }
1264
1265                    break;
1266            }
1267        default:
1268                glBegin( GL_POLYGON);   //create Polygon
1269                    glColor3f(      arrowColorR - Shadowing,
1270                                            arrowColorG - Shadowing,
1271                                            arrowColorB - Shadowing);
1272                    for(int i = 0; i < static_cast<int>(vBasePointsArrow.size()); i++) //run all
                            Base-Points-Koordinats
1273                    {
1274                                    glVertex3f( vBasePointsArrow[i].x + getPointX(0), //set Point
                                            of Polygon
1275                                                    vBasePointsArrow[i].y + getPointY(0),
1276                                                    vBasePointsArrow[i].z + getPointZ(0));
1277                    }
1278                glEnd();
1279                break;
1280
1281        }
1282
1283
1284
1285        return true;
1286 }
1287
1288
1289 ////////////////////////////////////////
1290 // get char* with lenght to next Curve
1291 //
1292 // needed for Output
1293 ////////////////////////////////////////
1294 char* Navi3D_Arrow::getNextCurveInMeter()
1295 {
1296        if(!isCurveOnStreet) //if no curve on street
1297                return (char*)"";
1298
1299        //calculate distance
1300
1301        double distance = getDistanceBetweenTwoNavPoints(0, firstCurveOnStreet);
1302
1303        //distance to int an write to stringstream
1304        stringstream tmp;
1305        tmp << static_cast<int>(distance);
1306
1307        //write output string
1308        string distanceString;
1309        tmp >> distanceString;
```

```
1310         distanceString+=" Meter ";
1311
1312         //set direction
1313         if(systemPar.KERNEL.interpolPoints.enable)
1314         {
1315                 if(getPointX(firstCurveOnStreet-systemPar.KERNEL.interpolPoints.resolution) <
                         getPointX(firstCurveOnStreet)  )
1316                         distanceString+=" <-";
1317                 else
1318                         distanceString+=" ->";
1319         }else{
1320                 if(getPointX(firstCurveOnStreet-1) < getPointX(firstCurveOnStreet)  )
1321                         distanceString+=" <-";
1322                 else
1323                         distanceString+=" ->";
1324 }
1325
1326
1327         //cast string to  char*
1328         char *stringOutput;
1329         stringOutput = new char[distanceString.length() + 1];
1330         strcpy(stringOutput, distanceString.c_str());
1331
1332         return stringOutput;
1333 }
1334
1335 ////////////////////////////////////////
1336 // get distance between to Navigation Points
1337 ////////////////////////////////////////
1338 double Navi3D_Arrow::getDistanceBetweenTwoNavPoints(int indexStart, int indexEnd)
1339 {
1340         double distance=0;
1341         for(int i=indexStart; i< indexEnd;i++)
1342         {
1343                 //Pytagoras aï¿œ+bï¿œ=cï¿œ
1344                 distance+=sqrt((getPointZ(i+1)-getPointZ(i))*(getPointZ(i+1)-getPointZ(i))+(getPointX(
                         i+1)-getPointX(i))*(getPointX(i+1)-getPointX(i)));
1345         }
1346         return distance;
1347 }
1348
1349 ////////////////////////////////////////
1350 // set Arrow Peak Type
1351 ////////////////////////////////////////
1352 bool Navi3D_Arrow::setArrowPeakType(int type)
1353 {
1354         if(type<1 || type>7)
1355         {
1356                 printLog("Navi3D_Arrow.cpp: ArrowPeakType not exist", 2);
1357                 return false;
1358         }
1359
1360         ArrowPeakType = type;
1361         return true;
1362
1363 }
1364
1365 ////////////////////////////////////////
1366 // set Arrow Begin Type
1367 ////////////////////////////////////////
1368 bool Navi3D_Arrow::setArrowBegin(int type)
1369 {
1370         if(type<1 || type>3)
1371         {
1372                 printLog("Navi3D_Arrow.cpp: Arrow begin not exist", 2);
1373                 return false;
1374         }
```

```
1375
1376        beginType = type;
1377        return true;
1378
1379    }
```

A.4.3 Interpolator: Headerdatei

```
1   ////////////////////////////////////////////////////////////////////////////////
2   //
3   //                      Hochschule Darmstadt - SS09 - Navi3D
4   //                              Copyright by h-da
5   //
6   //                      Created: Marco Muench / 2009/04/21
7   //
8   //                      Person in charge : Jens lorek
9   //
10  //                      Edit by:
11  //
12  //                      Iterpolate Points from GADA-Interface
13  //
14  ////////////////////////////////////////////////////////////////////////////////
15
16  #pragma once
17
18  /****
19   * Since the GADA interface provides only a limited number of points
20   * (avg. 3-6) per update, we interpolate between those key-positions to
21   * achieve a smooth curve. For an easy plug-an-play design of the components,
22   * the GADA-Interpolator (aka. Navi3D_InterpolatePoints) is designed as a
23   * wrapper for the 'real' GADA. Some calls are simply passed through and others
24   * - mostly switchBufferIfNew() - are modified to plug-in the interpolator
25   * passed in the constructor.
26   ****/
27
28  #include "../ARROW/IGenericInterpolator.h"
29  #include "../GADA/IGADA.h"
30  #include "../GADA/GADA.h"
31  #include "XML/Navi3D_Base.h"
32
33  class Navi3D_InterpolatePoints : public IGADA
34  {
35  public:
36          Navi3D_InterpolatePoints(GADA* gada, IGenericInterpolator* interpolator, N3D_System _systemPar
                ;
37          ~Navi3D_InterpolatePoints() { };
38
39          virtual int verticesCount();
40          virtual GAVertex getVertex(int index);
41          virtual int switchBufferIfNew();
42          virtual int getManoeuvrePoint();
43
44          virtual int getSpeed();
45          virtual int getYear();
46          virtual int getMonth();
47          virtual int getDay();
48          virtual int getUtc_hour();
49          virtual int getUtc_minute();
50          virtual int getUtc_second();
51          virtual int getLongitude();
52          virtual int getLatitude();
53          virtual int getHeight();
54          virtual int getHeading();
55
56          IGenericInterpolator* getInterpolator();
57          void setInterpolator(IGenericInterpolator* interpolator);
58
```

```
59 private:
60         IGenericInterpolator* m_Interpolator;
61         GADA* m_GADA;
62         N3D_System systemPar;
63 };
```

A.4.4 Interpolator: Sourcedatei

```
1  #include "ARROW/Navi3D_InterpolatePoints.h"
2  #include "ARROW/KBInterpolator.h"
3  #include "iostream"
4
5  Navi3D_InterpolatePoints::Navi3D_InterpolatePoints(GADA* gada, IGenericInterpolator* interpolator,
       N3D_System _systemPar)
6  :systemPar(_systemPar)
7  {
8          m_GADA = gada;
9          m_Interpolator = interpolator;
10 }
11
12 /////////////////
13 // Ugly stuff  // (caused by general architecture...)
14 /////////////////
15 IGenericInterpolator* Navi3D_InterpolatePoints::getInterpolator()
16 {
17         return m_Interpolator;
18 }
19
20 void Navi3D_InterpolatePoints::setInterpolator(IGenericInterpolator* interpolator)
21 {
22         m_Interpolator = interpolator;
23 }
24
25 ////////////////////////////////
26 // Pass calls to interpolator //
27 ////////////////////////////////
28 int Navi3D_InterpolatePoints::verticesCount()
29 {
30         return m_Interpolator->getNumVertices();
31 }
32
33 GAVertex Navi3D_InterpolatePoints::getVertex(int index)
34 {
35         return m_Interpolator->getVertex(index);
36 }
37
38 int Navi3D_InterpolatePoints::switchBufferIfNew()
39 {
40         int retVal = m_GADA->switchBufferIfNew();
41
42         if (retVal == 1)
43         {
44                 int numKeyPositions = m_GADA->verticesCount();
45                 m_Interpolator->resetInterpolator();
46
47                 if (numKeyPositions > 0)
48                 {
49                         for (int i = 0; i < numKeyPositions; i++)
50                                 m_Interpolator->addKeyPosition(m_GADA->getVertex(i));
51
52                         m_Interpolator->interpolate(systemPar.KERNEL.interpolPoints.resolution);
53                 }
54                 else
55                 {
56                         std::cout << "[*] Got ZERO points from GADA - wtf?!" << std::endl;
57                 }
58         }
```

```
59
60          return retVal;
61  }
62
63  ////////////////////////
64  // Pass calls to GADA //
65  ////////////////////////
66  int Navi3D_InterpolatePoints::getSpeed()
67  {
68          return m_GADA->getSpeed();
69  }
70
71  int Navi3D_InterpolatePoints::getYear()
72  {
73          return m_GADA->getYear();
74  }
75
76  int Navi3D_InterpolatePoints::getMonth()
77  {
78          return m_GADA->getMonth();
79  }
80
81  int Navi3D_InterpolatePoints::getDay()
82  {
83          return m_GADA->getDay();
84  }
85
86  int Navi3D_InterpolatePoints::getUtc_hour()
87  {
88          return m_GADA->getUtc_hour();
89  }
90
91  int Navi3D_InterpolatePoints::getUtc_minute()
92  {
93          return m_GADA->getUtc_minute();
94  }
95
96  int Navi3D_InterpolatePoints::getUtc_second()
97  {
98          return m_GADA->getUtc_second();
99  }
100
101 int Navi3D_InterpolatePoints::getLongitude()
102 {
103         return m_GADA->getLongitude();
104 }
105
106 int Navi3D_InterpolatePoints::getLatitude()
107 {
108         return m_GADA->getLatitude();
109 }
110
111 int Navi3D_InterpolatePoints::getHeight()
112 {
113         return m_GADA->getHeight();
114 }
115
116 int Navi3D_InterpolatePoints::getHeading()
117 {
118         return m_GADA->getHeading();
119 }
120
121 int Navi3D_InterpolatePoints::getManoeuvrePoint()
122 {
123         return m_GADA->getManoeuvrePoint();
124 }
```

A.4.5 KBInterpolator: Headerdatei

```
#pragma once

#include "ARROW/IGenericInterpolator.h"

class KBInterpolator : public IGenericInterpolator
{
	public:
		KBInterpolator();
		~KBInterpolator();

		virtual void addKeyPosition(GAVertex vertex);
		virtual void interpolate(int verticesPerSpline);
		virtual int getNumVertices();
		virtual GAVertex getVertex(int index);
		virtual void resetInterpolator();

		float getTension();
		float getBias();
		float getContinuity();

		void setTension(float value);
		void setBias(float value);
		void setContinuity(float value);

	private:
		GAVertex m_KeyPositions[128 + 2];
		GAVertex m_Vertices[128 * MAX_INTERPOLATION_RESOLUTION];

		int m_NumKeyPositions;
		int m_InterpolatedVertices;

		float m_Tension;
		float m_Bias;
		float m_Continuity;

		GAVertex cubicHermiteSplineInterpolation(GAVertex m1, GAVertex p1, GAVertex p2,
		    GAVertex m2, float t);
		GAVertex kochanekBartelsStartingTangent(GAVertex p0, GAVertex p1, GAVertex p2,
		    GAVertex p3);
		GAVertex kochanekBartelsEndingTangent(GAVertex p0, GAVertex p1, GAVertex p2, GAVertex
		    p3);
};
```

A.4.6 KBInterpolator: Sourcedatei

```
#include "ARROW/KBInterpolator.h"

KBInterpolator::KBInterpolator()
{
	m_InterpolatedVertices = 0;
	m_NumKeyPositions = 0;

	m_Tension = 0.6f;
	m_Bias = 0;
	m_Continuity = -0.2f;
}

KBInterpolator::~KBInterpolator()
{ }

void KBInterpolator::addKeyPosition(GAVertex vertex)
{
	m_KeyPositions[m_NumKeyPositions++] = vertex;
}

```

```
21 int KBInterpolator::getNumVertices()
22 {
23         return m_InterpolatedVertices;
24 }
25
26 GAVertex KBInterpolator::getVertex(int index)
27 {
28         return m_Vertices[index];
29 }
30
31 void KBInterpolator::resetInterpolator()
32 {
33         m_NumKeyPositions = 0;
34         m_InterpolatedVertices = 0;
35 }
36
37 void KBInterpolator::interpolate(int verticesPerSpline)
38 {
39         if (m_NumKeyPositions < 2)
40         {
41                 m_Vertices[0] = m_KeyPositions[0];
42                 m_InterpolatedVertices = 1;
43                 return;
44         }
45
46         if (verticesPerSpline > MAX_INTERPOLATION_RESOLUTION)
47                 verticesPerSpline = MAX_INTERPOLATION_RESOLUTION;
48
49         float tInc = 1.0f / (float)verticesPerSpline;
50         int numSegments = m_NumKeyPositions - 1;
51         m_InterpolatedVertices = 0;
52
53         GAVertex firstVertex = m_KeyPositions[0];
54         GAVertex lastVertex = m_KeyPositions[m_NumKeyPositions - 1];
55
56         for (int i = m_NumKeyPositions - 1; i >= 0; i--)
57                 m_KeyPositions[i + 1] = m_KeyPositions[i];
58
59         m_KeyPositions[0] = firstVertex;
60         m_KeyPositions[m_NumKeyPositions + 1] = lastVertex;
61
62         for (int segment = 0; segment < numSegments; segment++)
63         {
64                 GAVertex p0 = m_KeyPositions[segment];
65                 GAVertex p1 = m_KeyPositions[segment + 1];
66                 GAVertex p2 = m_KeyPositions[segment + 2];
67                 GAVertex p3 = m_KeyPositions[segment + 3];
68                 GAVertex startingTangent = kochanekBartelsStartingTangent(p0, p1, p2, p3);
69                 GAVertex endingTangent = kochanekBartelsEndingTangent(p0, p1, p2, p3);
70
71                 for (float t = 0; t < 1; t += tInc)
72                 {
73                         m_Vertices[m_InterpolatedVertices] = cubicHermiteSplineInterpolation(
                                startingTangent, p1, p2, endingTangent, t);
74                         m_InterpolatedVertices++;
75                 }
76         }
77 }
78
79 GAVertex KBInterpolator::cubicHermiteSplineInterpolation(GAVertex m1, GAVertex p1, GAVertex p2,
    GAVertex m2, float t)
80 {
81     float x = (2*t*t*t - 3*t*t + 1) * p1.x
82             + (t*t*t - 2*t*t + t) * m1.x
83             + (-2*t*t*t + 3*t*t) * p2.x
84             + (t*t*t - t*t) * m2.x;
85
```

```
86      float y = (2*t*t*t - 3*t*t + 1) * p1.y
87              + (t*t*t - 2*t*t + t) * m1.y
88              + (-2*t*t*t + 3*t*t) * p2.y
89              + (t*t*t - t*t) * m2.y;
90
91      float z = (2*t*t*t - 3*t*t + 1) * p1.z
92              + (t*t*t - 2*t*t + t) * m1.z
93              + (-2*t*t*t + 3*t*t) * p2.z
94              + (t*t*t - t*t) * m2.z;
95
96          GAVertex newVertex;
97          newVertex.x = x;
98          newVertex.y = y;
99          newVertex.z = z;
100         return newVertex;
101 }
102
103 GAVertex KBInterpolator::kochanekBartelsStartingTangent(GAVertex p0, GAVertex p1, GAVertex p2,
        GAVertex p3)
104 {
105     float x = ((1 - m_Tension) * (1 + m_Bias) * (1 + m_Continuity) / 2) * (p1.x - p0.x)
106             + ((1 - m_Tension) * (1 - m_Bias) * (1 - m_Continuity) / 2) * (p2.x - p1.x);
107
108     float y = ((1 - m_Tension) * (1 + m_Bias) * (1 + m_Continuity) / 2) * (p1.y - p0.y)
109             + ((1 - m_Tension) * (1 - m_Bias) * (1 - m_Continuity) / 2) * (p2.y - p1.y);
110
111         float z = ((1 - m_Tension) * (1 + m_Bias) * (1 + m_Continuity) / 2) * (p1.z - p0.z)
112             + ((1 - m_Tension) * (1 - m_Bias) * (1 - m_Continuity) / 2) * (p2.z - p1.z);
113
114         //* Starting tangent for Catmull-Rom interpolation
115         /*
116         float x = (p2.x - p0.x) / 2;
117         float y = (p2.y - p0.y) / 2;
118         float z = (p2.z - p0.z) / 2;
119         */
120
121         GAVertex startingTangent;
122         startingTangent.x = x;
123         startingTangent.y = y;
124         startingTangent.z = z;
125         return startingTangent;
126 }
127
128 GAVertex KBInterpolator::kochanekBartelsEndingTangent(GAVertex p0, GAVertex p1, GAVertex p2, GAVertex
        p3)
129 {
130     float x = ((1 - m_Tension) * (1 + m_Bias) * (1 - m_Continuity) / 2) * (p2.x - p1.x)
131             + ((1 - m_Tension) * (1 - m_Bias) * (1 + m_Continuity) / 2) * (p3.x - p2.x);
132
133     float y = ((1 - m_Tension) * (1 + m_Bias) * (1 - m_Continuity) / 2) * (p2.y - p1.y)
134             + ((1 - m_Tension) * (1 - m_Bias) * (1 + m_Continuity) / 2) * (p3.y - p2.y);
135
136     float z = ((1 - m_Tension) * (1 + m_Bias) * (1 - m_Continuity) / 2) * (p2.z - p1.z)
137             + ((1 - m_Tension) * (1 - m_Bias) * (1 + m_Continuity) / 2) * (p3.z - p2.z);
138
139         //* Ending tangent for Catmull-Rom interpolation
140         /*
141         float x = (p3.x - p1.x) / 2;
142         float y = (p3.y - p1.y) / 2;
143         float z = (p3.z - p3.z) / 2;
144         */
145
146         GAVertex endingTangent;
147         endingTangent.x = x;
148         endingTangent.y = y;
149         endingTangent.z = z;
150         return endingTangent;
```

```
151 }
152
153 //////////////
154 // GETTERS //
155 //////////////
156 float KBInterpolator::getTension()
157 {
158         return m_Tension;
159 }
160
161 float KBInterpolator::getBias()
162 {
163         return m_Bias;
164 }
165
166 float KBInterpolator::getContinuity()
167 {
168         return m_Continuity;
169 }
170
171 //////////////
172 // SETTERS //
173 //////////////
174 void KBInterpolator::setTension(float value)
175 {
176         m_Tension = value;
177 }
178
179 void KBInterpolator::setBias(float value)
180 {
181         m_Bias = value;
182 }
183
184 void KBInterpolator::setContinuity(float value)
185 {
186         m_Continuity = value;
187 }
```

A.4.7 IGenericInterpolator: Headerdatei

```
1  #pragma once
2
3  /****
4   * Interface for all types of interpolators.
5   * These can either be used in a stand-alone fashion
6   * or together with the Navi3D_InterpolatePoints class.
7   ****/
8
9  #include "../KERNEL/DataObjects.h"
10
11 class IGenericInterpolator
12 {
13         public:
14                 virtual ~IGenericInterpolator() {};
15
16                 // After intitializing the interpolator, all key-positions
17                 // must be added using this function
18                 virtual void addKeyPosition(GAVertex vertex) = 0;
19
20                 // By calling this function, the interpolation algorithm
21                 // is started and interpolates between all key-positions.
22                 // The vertices generated between each pair of key-positions
23                 // is controled by the second parameter and limited by the
24                 // MAX_INTERPOLATION_RESOLUTION constant
25                 virtual void interpolate(int verticesPerSpline) = 0;
26
27                 // After interpolation has finished, the total number of
```

```
28                  // vertices computed/availiable can be retrieved
29                  virtual int getNumVertices() = 0;
30
31                  // After interpolation has finished, each availiable vertex
32                  // can be retrieved using this function
33                  virtual GAVertex getVertex(int index) = 0;
34
35                  // To reuse the class-instance, all used member-variables
36                  // and the list of key-positions has to be reset
37                  virtual void resetInterpolator() = 0;
38  };
```

A.4.8 IGADA: Headerdatei

```
1   ////////////////////////////////////////////////////////////////////////////////
2   //
3   //                      Hochschule Darmstadt - SS09 - Navi3D
4   //                              Copyright by h-da
5   //
6   //                      Created: Jens Lorek / 2009/04/28
7   //
8   //                      Person in charge : Jens lorek
9   //
10  ////////////////////////////////////////////////////////////////////////////////
11
12  #pragma once
13
14  /****
15   * Interface for GADA. Since Navi3D_InterpolatePoints also implements
16   * this interface, it can act a wrapper for the 'real' GADA and be
17   * passed over to a Navi3D_Display instance.
18   ****/
19
20  #include "KERNEL/DataObjects.h"
21
22  class IGADA
23  {
24  public:
25          virtual ~IGADA() {};
26
27          virtual int switchBufferIfNew() = 0;
28          virtual int verticesCount() = 0;
29          virtual GAVertex getVertex(int index) = 0;
30          virtual int getManoeuvrePoint() = 0;
31
32          virtual int getSpeed() = 0;
33          virtual int getYear() = 0;
34          virtual int getMonth() = 0;
35          virtual int getDay() = 0;
36          virtual int getUtc_hour() = 0;
37          virtual int getUtc_minute() = 0;
38          virtual int getUtc_second() = 0;
39          virtual int getLongitude() = 0;
40          virtual int getLatitude() = 0;
41          virtual int getHeight() = 0;
42          virtual int getHeading() = 0;
43
44          GAVertex transform(GAVertex vertex)
45
46                  //transform from becker
47                  //if(systemPar.GADA.mode == BECKER_DEMO || systemPar.GADA.mode == STANDARD_ARROW)
48                  {
49                          Float32 x = -vertex.y;
50                          Float32 y = vertex.z;
51                          Float32 z = -vertex.x;
52
53                          vertex.x = x;
```

```
54                          vertex.y = y;
55                          vertex.z = z;
56                          return vertex;
57                  }
58
59                  //target
60                  GAVertex transformT(GAVertex vertex)
61                  {
62                          Float32 x = vertex.x;
63                          Float32 y = vertex.z;
64                          Float32 z = -vertex.y;
65
66                          vertex.x = x;
67                          vertex.y = y;
68                          vertex.z = z;
69                          return vertex;
70                  }
71  };
```

A.4.9 GADA: Headerdatei

```
1   //edit by Tobias Braun
2
3   #pragma once
4
5   #include "GADA/IGADA.h"
6
7   //icm socket zeug
8   #include "CInetAddr.h"
9   #include "CSockStream.h"
10  #include "CSockAcceptor.h"
11  #include "CSockConnector.h"
12  //#include <pthread.h>
13  //#include "KERNEL/CThread.h"
14  #include "KERNEL/CMutex.h"
15
16  #include "XML/Navi3D_Base.h"
17
18
19  //fuer windows api threads
20  #ifdef WINDOWS
21  #include <stdio.h>
22
23  #include <windows.h> // for all the Win32 specific thread information
24  DWORD WINAPI MyThread(LPVOID); // Standard function prototype for Win32 thread
25  #endif
26
27  typedef void * (*threadProc)(void *);
28
29  class GADA : public IGADA
30  {
31  public:
32          GADA(N3D_System tSystemPar);
33  //      GADA(N3D_System tSystemPar);
34          ~GADA();
35
36          bool connect();
37
38          virtual int verticesCount();
39          virtual GAVertex getVertex(int index);
40          virtual int switchBufferIfNew();
41          virtual int getManoeuvrePoint();
42
43          virtual int getSpeed();
44          virtual int getYear();
45          virtual int getMonth();
46          virtual int getDay();
```

```
47          virtual int getUtc_hour();
48          virtual int getUtc_minute();
49          virtual int getUtc_second();
50          virtual int getLongitude();
51          virtual int getLatitude();
52          virtual int getHeight();
53          virtual int getHeading();
54
55          volatile bool newGAData;
56          volatile bool newGPSData;
57
58
59  private:
60  #ifdef WINDOWS
61          //http://bytes.com/groups/net-vc/437827-running-class-method-thread-proc
62          static DWORD WINAPI ThreadStart(LPVOID info)
63                  {return static_cast<GADA*>(info)->Run();}
64          DWORD Run();
65  #else
66          static void ThreadStart(void* ptr);
67          static void ThreadStartT(void* ptr);
68          void Run();
69          void RunT();
70  #endif
71
72          //becker mode, the reason for this is not clear at all
73          GAHeader checkHeader;
74          //target mode
75          GAHeaderT checkHeaderT;
76
77          GAdata guidanceData[2];
78          volatile int readbufferGA,writebufferGA;
79
80          GPSdata gpsData[2];
81          volatile int readbufferGPS,writebufferGPS;
82
83          //own connection
84          CSockConnector myConnection;
85      CSockStream stream;
86          // Server Adrress
87          CInetAddr srvadr;
88  #ifdef WINDOWS
89          // thread id
90          DWORD threadID;
91          // thread handle
92          HANDLE thradHandle,mutexHandle;
93  #else
94
95          N3D_System systemPar;
96          //LINUX
97          pthread_t threadHandle;
98          CMutex mutexHandle;
99          int threadID;
100 #endif
101
102 };
```

A.4.10 GADA: Sourcedatei

```
1  #ifdef WINDOWS
2  #include "MS/tdAfx.h"
3  #endif
4  #include "GADA/GADA.h"
5  #include <iostream>
6  #include "KERNEL/Define.h"
7  //#include <pthread.h>
8
```

```
9
10 GADA::GADA(N3D_System tSystemPar)
11 :systemPar(tSystemPar)
12 {
13 ;
14         //for standard arrow it seems that the local ip is needed
15 if(systemPar.GADA.mode == STANDARD_ARROW)
16 {
17         this->srvadr.set(7000, "127.0.0.1");
18 }
19 else //becker demo and target
20 {
21         this->srvadr.set(systemPar.GADA.port, systemPar.GADA.IP.c_str());
22 }
23 #ifndef WINDOWS
24         //pthread_mutex_init(&mutexHandle, NULL); not needed any more ?!?
25 #endif
26 }
27 /*
28 GADA::GADA(N3D_System tSystemPar)
29 :systemPar(tSystemPar)
30 {
31         this->srvadr.set(tSystemPar.GADA.port, tSystemPar.GADA.IP.c_str());
32 #ifndef WINDOWS
33         //pthread_mutex_init(&mutexHandle, NULL); not needed any more ?!?
34 #endif
35 }*/
36
37 #ifndef WINDOWS
38 void GADA::ThreadStart(void* ptr){
39
40                 static_cast<GADA*>(ptr)->Run();
41 }
42 void GADA::ThreadStartT(void* ptr){
43                 static_cast<GADA*>(ptr)->RunT();
44 }
45 #endif
46
47 #ifdef WINDOWS
48 DWORD GADA::Run()
49 #else
50
51 void  GADA::Run()
52 #endif
53 {
54         while(1)
55         {
56 #ifdef WINDOWS
57                 WaitForSingleObject(mutexHandle,INFINITE);
58 #else
59                 mutexHandle.take();
60 #endif
61                 stream.recv(&checkHeader, sizeof(checkHeader));
62                 int stuff;
63
64                 //not used
65                 //int header = sizeof(gpsData[0].header);
66                 //int komp = sizeof(gpsData[0]);
67                 //int gps = sizeof(gpsData[0].data);
68
69
70                 if ( checkHeader.tag == 0x4741 )
71                 {
72                         stream.recv(&stuff, 2); // array bytes?!
73                         stream.recv(&guidanceData[writebufferGA].vertices, 2048);
74                         guidanceData[writebufferGA].header = checkHeader;
75                         newGAData=true;
```

```
76                  }
77                  else if ( checkHeader.tag == 0x4756 )
78                  {
79                          stream.recv(&gpsData[writebufferGPS].data, sizeof(gpsData[writebufferGPS])-
                                sizeof(GAHeader));
80                          gpsData[writebufferGPS].header = checkHeader;
81                          newGPSData=true;
82                  }
83
84                  //stream.recv(&guidanceData[writebufferGA], sizeof(guidanceData[writebufferGA]));
85                  //newGAData=true;
86
87  #ifdef WINDOWS
88                  ReleaseMutex(mutexHandle);
89  #else
90                  mutexHandle.give();
91  #endif
92
93
94
95          }
96  #ifdef WINDOWS
97          return 0;
98  #else
99          pthread_exit(NULL);
100
101 #endif
102 }
103
104 // for target mode
105 #ifdef WINDOWS
106 DWORD GADA::RunT()
107 #else
108
109 void  GADA::RunT()
110 #endif
111 {
112         //use not 1, keep possibility to cancel the thread
113         while(1)
114         {
115 #ifdef WINDOWS
116                 WaitForSingleObject(mutexHandle,INFINITE);
117 #else
118                 mutexHandle.take();
119 #endif
120                 stream.recv(&checkHeaderT, sizeof(checkHeaderT));
121                 int stuff;
122
123                 //not used
124                 //int header = sizeof(gpsData[0].header);
125                 //int komp = sizeof(gpsData[0]);
126                 //int gps = sizeof(gpsData[0].data);
127
128
129                 if ( checkHeaderT.tag == 0x4741 )
130                 {
131                         stream.recv(&stuff, 2); // array bytes?!
132                         stream.recv(&guidanceData[writebufferGA].vertices, 2048);
133                         guidanceData[writebufferGA].headerT = checkHeaderT;
134                         newGAData=true;
135                 }
136                 else if ( checkHeaderT.tag == 0x4756 )
137                 {
138                         stream.recv(&gpsData[writebufferGPS].data, sizeof(gpsData[writebufferGPS])-
                                sizeof(GAHeaderT));
139                         gpsData[writebufferGPS].headerT = checkHeaderT;
140                         newGPSData=true;
```

```
141                 }
142
143                 //stream.recv(&guidanceData[writebufferGA], sizeof(guidanceData[writebufferGA]));
144                 //newGAData=true;
145
146 #ifdef WINDOWS
147                 ReleaseMutex(mutexHandle);
148 #else
149                 mutexHandle.give();
150 #endif
151
152
153
154         }
155 #ifdef WINDOWS
156         return 0;
157 #else
158         pthread_exit(NULL);
159
160 #endif
161 }
162
163
164
165 int GADA::switchBufferIfNew()
166 {
167         if(!newGAData)
168         {
169                 if(!newGPSData)
170                         return 0;
171                 else
172                 {
173 #ifdef WINDOWS
174                 WaitForSingleObject(mutexHandle,INFINITE);
175 #else //LINUX
176                 //this leads to very bad performance under linux (diaoshow of the arow)
177                 //readbufferGPS and wirtebufferGPS are now volatile to make sure, no problems occur
                        here
178                 //mutexHandle.take();
179 #endif
180                         int tmp = readbufferGPS;
181                         readbufferGPS = writebufferGPS;
182                         writebufferGPS = tmp;
183                         newGPSData = false;
184 #ifdef WINDOWS
185                         ReleaseMutex(mutexHandle);
186 #else // LINUX
187                         //mutexHandle.give();
188 #endif
189
190                         return 2;
191                 }
192         }
193         else
194         {
195 #ifdef WINDOWS
196                 WaitForSingleObject(mutexHandle,INFINITE);
197 #else //LINUX
198                 //this leads to very bad performance under linux (diaoshow of the arow)
199                 //readbufferGPS and wirtebufferGPS are now volatile to make sure, no problems occur
                        here
200                 //mutexHandle.take();
201 #endif
202                 int tmp = readbufferGA;
203                 readbufferGA = writebufferGA;
204                 writebufferGA = tmp;
205                 newGAData = false;
```

```
206 #ifdef WINDOWS
207                 ReleaseMutex(mutexHandle);
208 #else //LINUX
209                 //mutexHandle.give();
210 #endif
211
212                 return 1;
213         }
214 }
215
216 bool GADA::connect()
217 {
218         //open connection
219         bool retVal = myConnection.connect(stream, srvadr, false);
220         if(retVal)
221         {
222                 //some init stuff
223                 newGAData = false;
224                 writebufferGA=1;
225                 readbufferGA=0;
226                 //some init stuff
227                 newGPSData = false;
228                 writebufferGPS=1;
229                 readbufferGPS=0;
230
231 #ifdef WINDOWS
232                 //create mutex and thread
233                 mutexHandle = CreateMutex(NULL,FALSE,NULL);
234                 thradHandle = CreateThread(NULL,0,&GADA::ThreadStart,this,NULL,&threadID);
235 #else //LINUX
236                 threadID = 0;
237
238                 //pthread_create(&threadHandle, NULL, (threadProc) GADA::Run ,NULL);
239         //      pthread_create(&threadHandle, NULL, (threadProc) GADA::ThreadStart ,this);
240                 if(systemPar.GADA.mode == BECKER_DEMO || systemPar.GADA.mode == STANDARD_ARROW)
241                 {
242                         pthread_create(&threadHandle, NULL, (threadProc) GADA::ThreadStart, this);
243                 }
244                 else
245                 {
246                         pthread_create(&threadHandle, NULL, (threadProc) GADA::ThreadStartT, this);
247
248                 }
249
250 #endif
251
252
253
254         }
255
256         //20090526 if no connection is found, this must be initialized
257         writebufferGA=1;
258         readbufferGA=0;
259
260         return retVal;
261 }
262
263 GADA::~GADA(void)
264 {
265
266
267
268 #ifdef WINDOWS
269         //hier sollte noch der thrad abgesch(l)ossen werden
270         CloseHandle(mutexHandle);
271 #else //LINUX
272         mutexHandle.~CMutex();
```

```
273         pthread_exit(NULL);
274 #endif
275 }
276
277 int GADA::verticesCount()
278 {
279         if(systemPar.GADA.mode == STANDARD_ARROW )
280         {
281                 return 8;
282         }
283         else
284         {
285                 return guidanceData[readbufferGA].header.numberOfVertices;
286         }
287 }
288
289
290
291 GAVertex GADA::getVertex(int index)
292 {
293         GAVertex temp;
294         temp.x = 0;
295         temp.y = 0;
296         temp.z = 0;
297
298         if(systemPar.GADA.mode == STANDARD_ARROW )
299         {
300
301                 if(index == 0)
302                 {
303                         temp.x = 0;
304                         temp.y = 0;
305                         temp.z = 0;
306                         return temp;
307                 }
308                 else if( index == 1)
309                 {
310                         temp.x = 0;
311                         temp.y = 0;
312                         temp.z = -5;
313                         return temp;
314                 }
315                 else if( index == 2)
316                 {
317                         temp.x = 8;
318                         temp.y = 0;
319                         temp.z = -5;
320                         return temp;
321                 }
322                 else if( index == 3)
323                 {
324                         temp.x = 10;
325                         temp.y = 0;
326                         temp.z = -10;
327                         return temp;
328                 }
329                 else if( index == 4)
330                 {
331                         temp.x = 5;
332                         temp.y = 0;
333                         temp.z = -15;
334                         return temp;
335                 }
336                 else if( index == 5)
337                 {
338                         temp.x = 5;
339                         temp.y = 0;
```

```
340                         temp.z = -20;
341                         return temp;
342                 }
343                 else if( index == 6)
344                 {
345                         temp.x = -10;
346                         temp.y = 0;
347                         temp.z = -20;
348                         return temp;
349                 }
350                 else if( index == 7)
351                 {
352                         temp.x = -20;
353                         temp.y = 0;
354                         temp.z = -20;
355                         return temp;
356                 }
357                 else if( index == 8)
358                 {
359                         temp.x = -15;
360                         temp.y = 0;
361                         temp.z = -25;
362                         return temp;
363                 }
364                 else //to clear out warning message, should never occour
365                 {
366                         temp.x = 0;
367                         temp.y = 0;
368                         temp.z = 0;
369                         return temp;
370                 }
371         }
372         else
373         {
374
375                 if(systemPar.GADA.mode == BECKER_DEMO || systemPar.GADA.mode == STANDARD_ARROW)
376                         return transform(guidanceData[readbufferGA].vertices[index]);
377                 else
378                         return transformT(guidanceData[readbufferGA].vertices[index]);
379         }
380 }
381
382 int GADA::getManoeuvrePoint()
383 {
384         return guidanceData[readbufferGA].header.manPointIndex;
385 }
386
387 int GADA::getSpeed()
388 {
389         return gpsData[readbufferGPS].data.speed;
390 }
391
392 int GADA::getYear()
393 {
394         return gpsData[readbufferGPS].data.year;
395 }
396
397 int GADA::getMonth()
398 {
399         return gpsData[readbufferGPS].data.year;
400 }
401
402 int GADA::getDay()
403 {
404         return gpsData[readbufferGPS].data.year;
405 }
406
```

```
407 int GADA::getUtc_hour()
408 {
409         return gpsData[readbufferGPS].data.utc_hour;
410 }
411
412 int GADA::getUtc_minute()
413 {
414         return gpsData[readbufferGPS].data.utc_minute;
415 }
416
417 int GADA::getUtc_second()
418 {
419         return gpsData[readbufferGPS].data.utc_second;
420 }
421
422 int GADA::getLongitude()
423 {
424         return gpsData[readbufferGPS].data.longitude;
425 }
426
427 int GADA::getLatitude()
428 {
429         return gpsData[readbufferGPS].data.latitude;
430 }
431
432 int GADA::getHeight()
433 {
434         return gpsData[readbufferGPS].data.height;
435 }
436
437 int GADA::getHeading()
438 {
439         return gpsData[readbufferGPS].data.heading;
440 }
```

A.4.11 Navi3D_Display: Headerdatei

```
1  ////////////////////////////////////////////////////////////////////////////////
2  //
3  //                      Hochschule Darmstadt - SS09 - Navi3D
4  //                              Copyright by h-da
5  //
6  //                      Created: Marco Muench / 2009/04/21
7  //
8  //                      Person in charge : Marco Muench, Michael Roth
9  //
10 //                      last Edit by: Marco Muench, 24.6.2009
11 //
12 //                      OpenGL Navi3D-Application
13 //
14 ////////////////////////////////////////////////////////////////////////////////
15
16 #pragma once
17 #include "ARROW/Navi3D_InterpolatePoints.h"
18 #include "XML/Navi3D_Base.h"
19 #include "ARROW/Navi3D_SunPos.h"
20 #include "POI/Navi3D_POI.h"
21 #include "ARROW/Navi3D_Arrow.h"
22 #include "KERNEL/Define.h"
23 #include "FAHR/Navi3D_Fahrspur.h"
24 #include "GIMMICK/Navi3D_GIMMICK.h"
25
26 //#include <windows.h>
27 #include <stdio.h>
28 //#include <GL/glut.h>
29 #include <iostream>
30 #include <time.h>
```

```
//#include <GL/glx.h>
//#include <GL/glext.h>
//#include <GL/glxext.h>
#ifndef M_PI
#define M_PI    3.14159265358979323846f
#endif

#ifdef SHADER
#include "Shader/Shader.h"
#endif

using namespace std;

class Navi3D_Display{
public:
        Navi3D_Display(void);
        Navi3D_Display(int argc, char** argv, IGADA* _pGADA, Navi3D_Base* _pBase);
        ~Navi3D_Display(void);
        void Geometry();
        void switchBufferIfNew();

        float eyeX, eyeY, eyeZ;
        float Augenabstand;

        void changeArrowColor(int newColor);
        void ArrowSegmentation();
        void setNewArrowShadow(float newMaxShadow);
        void setNewBaseFactor(float factor);
        void setNewArrowProfile(int profileNr);
        void setArrowPeakType(int ArrowPeakType);
        void setArrowBegin(int ArrowBegin);
        void calculateArrow();

private:
        IGADA* pGADA;
        Navi3D_Base *pBase;
        N3D_System systemPar;
        Navi3D_SunPos *pSunpos;
        Navi3D_POI *pPOI;
        Navi3D_Arrow *pArrow;
        Navi3D_Fahrspur *pFahrspur;
        Navi3D_GIMMICK *pGIMMICK;
        void init(void);
};
```

A.4.12 Navi3D_Display: Sourcedatei

```
#include "Navi3D_Display.h"
#include "ARROW/KBInterpolator.h"
#include "Shader/Textures.h"
#include "Shader/ShadowObjekts.h"


extern void printLog(string Log, unsigned int Level);

#define red 0
#define green 1
#define blue 2
#define pink 3
#define yellow 4
#define turquoise 5
#define white 6

int font=(int)GLUT_BITMAP_TIMES_ROMAN_24;
int bitmapHeight=25;
int winwidth;
int winheight;
```

```
21
22  #ifdef TIMER
23          clock_t oldTime = 0, newTime = 0;
24          int functionTimer =0;
25  #endif
26
27
28  void setOrthographicProjection() {
29
30          winwidth = glutGet(GLUT_WINDOW_WIDTH);
31          winheight = glutGet(GLUT_WINDOW_HEIGHT);
32
33          // switch to projection mode
34          glMatrixMode(GL_PROJECTION);
35          // save previous matrix which contains the
36          //settings for the perspective projection
37          glPushMatrix();
38          // reset matrix
39          glLoadIdentity();
40          // set a 2D orthographic projection
41          gluOrtho2D(0, winwidth, 0, winheight);
42          // invert the y axis, down is positive
43          glScalef(1, -1, 1);
44          // mover the origin from the bottom left corner
45          // to the upper left corner
46          glTranslatef(0, -winheight, 0);
47          glMatrixMode(GL_MODELVIEW);
48  }
49
50  void resetPerspectiveProjection() {
51          glMatrixMode(GL_PROJECTION);
52          glPopMatrix();
53          glMatrixMode(GL_MODELVIEW);
54  }
55
56  void renderBitmapString(float x, float y, void *font,char *string)
57  {
58
59    char *c;
60    glRasterPos2f(x, y);
61    for (c=string; *c != '\0'; c++) {
62      glutBitmapCharacter(font, *c);
63    }
64  }
65
66  void renderSpacedBitmapString(float x, float y,int spacing, void *font,char *string) {
67    char *c;
68    int x1=x;
69    for (c=string; *c != '\0'; c++) {
70          glRasterPos2f(x1,y);
71      glutBitmapCharacter(font, *c);
72          x1 = x1 + glutBitmapWidth(font,*c) + spacing;
73    }
74  }
75
76
77  void renderVerticalBitmapString(float x, float y, int bitmapHeight, void *font,char *string)
78  {
79
80    char *c;
81    int i;
82    for (c=string,i=0; *c != '\0'; i++,c++) {
83          glRasterPos2f(x, y+bitmapHeight*i);
84      glutBitmapCharacter(font, *c);
85    }
86  }
87
```

```
88
89
90  Navi3D_Display *pDisplay;
91  Navi3D_GIMMICK *pGIMMICK_global;
92
93  //* Woohaaa - This is DEBUG ONLY shit...
94  Navi3D_InterpolatePoints* glutGADA;
95  KBInterpolator* glutInterpolator;
96
97  //Glut Lightsources
98  GLfloat LightAmbient[]= { 0.2f, 0.2f, 0.2f, 1.0f };  //Ambient of Lightsource
99  GLfloat LightDiffuse[]= { 1.0f, 0.5f, 1.0f, 1.0f };      //Diffuse of Lightsource
100 GLfloat LightPosition[]= { 0.0f, 0.0f, 0.0f, 1.0f };    //Position of Lightsource
101 GLfloat color[4]= {0.0f,0.0f,0.0f,0.0f};
102
103 float lightposition[4]={ 0.0f, 0.0f, 0.0f, 1.0f };
104
105
106 void standard(int id)
107 { switch (id) {
108         case 100: exit(0); break;
109         ///change Arrow Color
110         case 201: pDisplay->changeArrowColor(red); break;
111         case 202: pDisplay->changeArrowColor(green); break;
112         case 203: pDisplay->changeArrowColor(blue); break;
113         case 204: pDisplay->changeArrowColor(yellow); break;
114         case 205: pDisplay->changeArrowColor(turquoise); break;
115         case 206: pDisplay->changeArrowColor(pink); break;
116         case 207: pDisplay->changeArrowColor(white); break;
117         //change Arrow properties
118         case 301: pDisplay->ArrowSegmentation(); break;
119         case 310: pDisplay->setNewArrowShadow(0.0); break;
120         case 311: pDisplay->setNewArrowShadow(0.25); break;
121         case 312: pDisplay->setNewArrowShadow(0.5); break;
122         case 313: pDisplay->setNewArrowShadow(0.75); break;
123         case 314: pDisplay->setNewArrowShadow(1.0); break;
124         //change Base properties
125         case 320: pDisplay->setNewBaseFactor(1.0); break;
126         case 321: pDisplay->setNewBaseFactor(1.5); break;
127         case 322: pDisplay->setNewBaseFactor(2.0); break;
128         case 323: pDisplay->setNewBaseFactor(2.5); break;
129         //set new Arrow Profile
130         case 330: pDisplay->setNewArrowProfile(1); break;
131         case 331: pDisplay->setNewArrowProfile(2); break;
132         case 332: pDisplay->setNewArrowProfile(3); break;
133         case 333: pDisplay->setNewArrowProfile(4); break;
134         case 334: pDisplay->setNewArrowProfile(5); break;
135         //set new Arrow Peak Type
136         case 340: pDisplay->setArrowPeakType(1); break;
137         case 341: pDisplay->setArrowPeakType(2); break;
138         case 342: pDisplay->setArrowPeakType(3); break;
139         case 343: pDisplay->setArrowPeakType(4); break;
140         case 344: pDisplay->setArrowPeakType(5); break;
141         case 345: pDisplay->setArrowPeakType(6); break;
142         case 346: pDisplay->setArrowPeakType(7); break;
143         //set new Arrow begin
144         case 350: pDisplay->setArrowBegin(1); break;
145         case 351: pDisplay->setArrowBegin(2); break;
146         case 352: pDisplay->setArrowBegin(3); break;
147
148         }
149 }
150
151 /* Menu registrieren */
152 void mouseMenu(void)
153 {
154         int menu, subMenuArrowColor, subMenuDebug, subMenuArrow,
```

```
155                 subMaxShadow, subBaseFactor, subArrowProfile, subArrowPeakType,
156                 subArrowBegin;
157
158         subMenuArrowColor = glutCreateMenu(standard);
159         glutAddMenuEntry("red",201);
160         glutAddMenuEntry("green",202);
161         glutAddMenuEntry("blue",203);
162         glutAddMenuEntry("yellow",204);
163         glutAddMenuEntry("turquoise",205);
164         glutAddMenuEntry("pink",206);
165         glutAddMenuEntry("white",207);
166
167         subMaxShadow = glutCreateMenu(standard);
168         glutAddMenuEntry("0.00 (no Shadow)",310);
169         glutAddMenuEntry("0.25 ",311);
170         glutAddMenuEntry("0.50 (medium Shadow)",312);
171         glutAddMenuEntry("0.75",313);
172         glutAddMenuEntry("1.00 (max. Shadow)",314);
173
174         subMenuArrow = glutCreateMenu(standard);
175         glutAddMenuEntry("Arrowsegmentation",301);
176
177         subBaseFactor = glutCreateMenu(standard);
178         glutAddMenuEntry("Factor = 1.0",320);
179         glutAddMenuEntry("Factor = 1.5",321);
180         glutAddMenuEntry("Factor = 2.0",322);
181         glutAddMenuEntry("Factor = 2.5",323);
182
183         subArrowProfile = glutCreateMenu(standard);
184         glutAddMenuEntry("Profile 1",330);
185         glutAddMenuEntry("Profile 2",331);
186         glutAddMenuEntry("Profile 3",332);
187         glutAddMenuEntry("Profile 4",333);
188         glutAddMenuEntry("Profile 5",334);
189
190         subArrowPeakType= glutCreateMenu(standard);
191         glutAddMenuEntry("Normale Pfeilspitze 1",340);
192         glutAddMenuEntry("Normale Pfeilspitze 2",341);
193         glutAddMenuEntry("Normale Pfeilspitze 3",342);
194         glutAddMenuEntry("Normale Pfeilspitze 4",343);
195         glutAddMenuEntry("2D-Spitze",344);
196         glutAddMenuEntry("Bessere Pfeilspitze 1",345);
197         glutAddMenuEntry("Bessere Pfeilspitze 2",346);
198
199         subArrowBegin= glutCreateMenu(standard);
200         glutAddMenuEntry("Start Normal 1",350);
201         glutAddMenuEntry("Start vom Boden 2",351);
202         glutAddMenuEntry("Abgerundet 3",352);
203
204
205         subMenuDebug = glutCreateMenu(standard);
206         glutAddSubMenu("Arrow",subMenuArrow);
207
208         menu = glutCreateMenu(standard);
209         glutAddSubMenu("DEBUG",subMenuDebug);
210         glutAddSubMenu("Arrowcolor",subMenuArrowColor);
211         glutAddSubMenu("Shadow",subMaxShadow);
212         glutAddSubMenu("Base Factor",subBaseFactor);
213         glutAddSubMenu("Profile",subArrowProfile);
214         glutAddSubMenu("Pfeilstart",subArrowBegin);
215         glutAddSubMenu("Pfeilspitze",subArrowPeakType);
216         glutAddMenuEntry("close",100);
217         glutAttachMenu(GLUT_RIGHT_BUTTON);
218 }
219
220
221
```

```
222 #ifdef SHADER
223         /// Shader
224         GLhandleARB shader_Pfeil; //shader program
225         GLhandleARB VS_Pfeil; //vertex shader
226         GLhandleARB PS_Pfeil; //pixel shader
227 
228         GLhandleARB shader_Steinboden; //shader program
229         GLhandleARB VS_Steinboden; //vertex shader
230         GLhandleARB PS_Steinboden; //pixel shader
231 
232         GLhandleARB shader_Sonne; //shader program
233         GLhandleARB VS_Sonne; //vertex shader
234         GLhandleARB PS_Sonne; //pixel shader
235 
236         // Methoden für Shadowmap
237         void shadowBegin(); //Beginn eines Shadowmap Model
238         void shadowEnd();   //Ende eines Shadowmap Model
239         //void bauSchattenSzene(); //Baut die Szene für die Shadowmap
240         float proj[16];
241         float mv[16];
242         GLuint depthTex; //Datei um die Shadowmap als Textur zu speichern
243 
244 
245         //Extensions für Multitexturing
246         //PFNGLACTIVETEXTUREPROC        glActiveTexture; //Active Texture setzt aktive Textureinheit
247         //PFNGLMULTITEXCOORD2FPROC glMultiTexCoord2f; //setzt 2d-texturkoordinaten einer textureinheit
248         //PFNGLMULTITEXCOORD3FPROC glMultiTexCoord3f; //setzt 3d-texturkoordinaten einer textureinheit
249 
250         //Extension functions Shader
251         extern PFNGLCREATEPROGRAMOBJECTARBPROC          glCreateProgramObjectARB;
252         extern PFNGLCREATESHADEROBJECTARBPROC       glCreateShaderObjectARB;
253         extern PFNGLSHADERSOURCEARBPROC                 glShaderSourceARB;
254         extern PFNGLCOMPILESHADERARBPROC                        glCompileShaderARB;
255         extern PFNGLATTACHOBJECTARBPROC                 glAttachObjectARB;
256         extern PFNGLLINKPROGRAMARBPROC                          glLinkProgramARB;
257         extern PFNGLUSEPROGRAMOBJECTARBPROC             glUseProgramObjectARB;
258         extern PFNGLDELETEOBJECTARBPROC                 glDeleteObjectARB;
259         extern PFNGLUNIFORM1IARBPROC                            glUniform1iARB;
260         extern PFNGLUNIFORM1FARBPROC                            glUniform1fARB;
261         extern PFNGLUNIFORM4FARBPROC                            glUniform4fARB;
262         extern PFNGLUNIFORM4FVARBPROC                           glUniform4fvARB;
263         extern PFNGLGETUNIFORMLOCATIONARBPROC           glGetUniformLocationARB;
264         extern PFNGLVERTEXATTRIB4FARBPROC                       glVertexAttrib4fARB     ;
265         extern PFNGLGETATTRIBLOCATIONARBPROC            glGetAttribLocationARB;
266         extern PFNGLGETOBJECTPARAMETERIVARBPROC glGetObjectParameterivARB;
267         extern PFNGLGETINFOLOGARBPROC                           glGetInfoLogARB;
268         extern PFNGLVALIDATEPROGRAMARBPROC                      glValidateProgramARB;
269 
270         GLuint texture[9];
271 
272 
273 
274 void shadowBegin()
275 {
276         glActiveTexture(GL_TEXTURE1);
277         glMatrixMode(GL_TEXTURE);
278         glLoadIdentity();
279         glMatrixMode(GL_MODELVIEW);
280         glActiveTexture(GL_TEXTURE2);
281         glBindTexture(GL_TEXTURE_2D, depthTex);
282         glMatrixMode(GL_TEXTURE);
283         glLoadIdentity();
284         glTranslatef(0.5f,0.5f,0.5f);
285         glScalef(0.5f,0.5f,0.5f);
286         glMultMatrixf(proj);
287         glMultMatrixf(mv);
288         glMatrixMode(GL_MODELVIEW);
```

```
289 }
290
291 void shadowEnd()
292 {
293         glActiveTexture(GL_TEXTURE2);
294         glMatrixMode(GL_TEXTURE);
295         glLoadIdentity();
296         glMatrixMode(GL_MODELVIEW);
297         glActiveTexture(GL_TEXTURE0);
298 }
299 #endif
300
301
302 void Navi3D_Display::switchBufferIfNew(){
303         pGADA->switchBufferIfNew();
304 }
305
306 void keyboard ( unsigned char key, int x, int y )  // Create Keyboard Function
307 {
308         float newValue = 0;
309
310   switch ( key ) {
311
312     case 27:        // When Escape Is Pressed...
313       exit ( 0 );   // Exit The Program
314       break;        // Ready For Next Case
315
316         //////////////////////////Speichern
317
318         case 98: //b-Taste
319         case 65:
320            //EinstellungSpeichern();
321         break;
322
323         ////////////////////////Pfeil
324
325         case 97: //a-Taste
326         case 64:
327            //Pfeilbreite = Pfeilbreite-0.01;
328         break;
329
330         case 81://q-Taste
331         case 113:
332            //Pfeilbreite = Pfeilbreite+0.01;
333         break;
334
335         case 115: //s-Taste
336         case 83:
337            //Pfeilhoehe = Pfeilhoehe-0.01;
338         break;
339
340         case 87://w-Taste
341         case 119:
342            //Pfeilhoehe = Pfeilhoehe+0.01;
343         break;
344
345         ////////////////////////Bildschirm
346
347         case 77: //m-Taste
348         case 109:
349            glutFullScreen ( );
350         break;
351
352         case 78://n-Taste
353         case 110:
354            glutReshapeWindow ( 500, 500 );
355         break;
```

```
356
357         ////////////////////////Augenabstand
358
359         case 90: //z-Taste
360         case 122:
361            pDisplay->Augenabstand = pDisplay->Augenabstand + 0.005;
362         break;
363
364         case 85://u-Taste
365         case 117:
366            pDisplay->Augenabstand = pDisplay->Augenabstand - 0.005;
367         break;
368
369         ////////////////////////Farben
370
371         case 69: //e-Taste
372         case 101:
373                 pGIMMICK_global->erhoeheGeschwindigkeit(1);
374
375         break;
376
377         case 68://d-Taste
378         case 100:
379                 pGIMMICK_global->verringereGeschwindigkeit(1);
380
381         break;
382
383         case 82: //r-Taste
384         case 114:
385
386         break;
387
388         case 70://f-Taste
389         case 102:
390
391         break;
392
393         case 84: //t-Taste
394         case 116:
395
396         break;
397
398         case 71://g-Taste
399         case 103:
400
401         break;
402
403         /////////// Vordef. Farben
404
405         case 89://y-Taste
406         case 121:
407
408         break;
409
410         case 88: //x-Taste
411         case 120:
412
413         break;
414
415         case 67://c-Taste
416         case 99:
417
418         break;
419
420 //* Interpolator fine tuning
421 //#ifdef INTERPOLATE_POINTS
422         //* i - Tension++
```

```
423         case 105:
424                 newValue = glutInterpolator->getTension() + 0.1f;
425                 glutInterpolator->setTension(newValue);
426                 cout << "[*] Tension = " << newValue << endl;
427                 break;
428 
429         //* j - Tension--
430         case 106:
431                 newValue = glutInterpolator->getTension() - 0.1f;
432                 glutInterpolator->setTension(newValue);
433                 cout << "[*] Tension = " << newValue << endl;
434                 break;
435 
436         //* o - Bias++
437         case 111:
438                 newValue = glutInterpolator->getBias() + 0.1f;
439                 glutInterpolator->setBias(newValue);
440                 cout << "[*] Bias = " << newValue << endl;
441                 break;
442 
443         //* k - Bias--
444         case 107:
445                 newValue = glutInterpolator->getBias() - 0.1f;
446                 glutInterpolator->setBias(newValue);
447                 cout << "[*] Bias = " << newValue << endl;
448                 break;
449 
450         //* p - Continuity++
451         case 112:
452                 newValue = glutInterpolator->getContinuity() + 100.1f;
453                 glutInterpolator->setContinuity(newValue);
454                 cout << "[*] Continuity = " << newValue << endl;
455                 break;
456 
457         //* l - Continuity--
458         case 108:
459                 newValue = glutInterpolator->getContinuity() - 0.1f;
460                 glutInterpolator->setContinuity(newValue);
461                 cout << "[*] Continuity = " << newValue << endl;
462                 break;
463 //#endif
464 
465     default:        // Now Wrap It Up
466       break;
467   }
468 }
469 float demoHead = 0.0;
470 float timeON = false;
471 void arrow_keys ( int a_keys, int x, int y )  // Create Special Function (required for arrow keys)
472 {
473   switch ( a_keys ) {
474     case GLUT_KEY_UP:     // When Up Arrow Is Pressed...
475                 pDisplay->eyeY = pDisplay->eyeY+0.1;
476       break;
477     case GLUT_KEY_DOWN:               // When Down Arrow Is Pressed...
478                 pDisplay->eyeY = pDisplay->eyeY-0.1;
479       break;
480         case GLUT_KEY_PAGE_UP:
481                 pDisplay->eyeZ = pDisplay->eyeZ  -0.1;
482                 break;
483         case GLUT_KEY_PAGE_DOWN:
484                 pDisplay->eyeZ = pDisplay->eyeZ  +0.1;
485                 break;
486         case GLUT_KEY_LEFT:
487                 pDisplay->eyeX = pDisplay->eyeX-0.1;
488       break;
489     case GLUT_KEY_RIGHT:
```

```
                pDisplay->eyeX = pDisplay->eyeX+0.1;
      break;
        case GLUT_KEY_F1:
                demoHead+=90;
      break;
        case GLUT_KEY_F2:
                {
                        if(timeON)
                                timeON=false;
                        else
                                timeON=true;
                }
      break;
        case GLUT_KEY_END:
                break;
    default:
      break;
  }
}


void display( void )
{
        pDisplay->switchBufferIfNew();
        pDisplay->calculateArrow();

        pGIMMICK_global->calculateRouteData();

        winwidth = glutGet(GLUT_WINDOW_WIDTH);
        winheight = glutGet(GLUT_WINDOW_HEIGHT);


#ifdef STEREOPROJECTION

        glLoadIdentity ( );


        glMatrixMode (GL_PROJECTION);
        glLoadIdentity();

        gluPerspective(45.0f,(GLfloat)(winwidth/2.0f)/
                        (GLfloat)winheight,0.1f, 3000.0f);

        // Szene fuer das linke Auge rendern (linker Viewport)
        // linke haelfte des Fensters
        glViewport (0, 0, winwidth/2,winheight);

        // Matrix sichern (fuer das rechte Auge)
        glPushMatrix();

        //glLoadIdentity ( );
                gluLookAt(pDisplay->eyeX+pDisplay->Augenabstand, pDisplay->eyeY, pDisplay->eyeZ,
                      0+pDisplay->Augenabstand, 0, -10,
                          0.0f,1.0f,0.0f);

        glMatrixMode (GL_MODELVIEW);
        glLoadIdentity();

        // Framebuffer,Depthbuffer loeschen
        glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        pDisplay->Geometry();

                // Szene fuer das rechte Auge rendern (rechter Viewport)
        glViewport (winwidth/2, 0, winwidth/2, winheight);
        glMatrixMode (GL_PROJECTION);

```

```
557         // WICHTIG:
558         // Kein glClear aufrufen da sonst die linke Szene
559         // verlohren geht
560         // Matrix vom Stack holen
561         glPopMatrix();
562 
563         // Kamera fuer das rechte Auge verschieben
564         //gluLookAt(auge_d/2.0,0.0,1.0, auge_d/2.0,0.0,0.0,0.0,1.0,0.0);
565         gluLookAt(pDisplay->eyeX-pDisplay->Augenabstand, pDisplay->eyeY, pDisplay->eyeZ,
566                   0-pDisplay->Augenabstand, 0, -10,
567                       0.0f,1.0f,0.0f);
568 
569 
570         glMatrixMode (GL_MODELVIEW);
571         glLoadIdentity();
572 
573         // Aufrufen der Szenen Geometrie
574         pDisplay->Geometry();
575 
576         glutSwapBuffers();
577 #else
578         glPushMatrix();
579         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
580 
581         glLoadIdentity ( );
582 
583 
584         glMatrixMode (GL_PROJECTION);
585         glLoadIdentity();
586 
587         gluPerspective(45.0f,(GLfloat)(winwidth)/
588                        (GLfloat)winheight,0.1f, 3000.0f);
589 
590         // Szene fuer das linke Auge rendern (linker Viewport)
591         // linke haelfte des Fensters
592         glViewport (0, 0, winwidth,winheight);
593 
594         // Matrix sichern (fuer das rechte Auge)
595         glPushMatrix();
596 
597         //glLoadIdentity ( );
598                 gluLookAt(pDisplay->eyeX+pDisplay->Augenabstand, pDisplay->eyeY, pDisplay->eyeZ,
599                       0+pDisplay->Augenabstand, 0, -10,
600                           0.0f,1.0f,0.0f);
601 
602         glMatrixMode (GL_MODELVIEW);
603         glLoadIdentity();
604 
605         // Framebuffer,Depthbuffer loeschen
606         glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
607 
608         pDisplay->Geometry();
609 
610         glutSwapBuffers();
611 
612 #endif
613 
614 
615        #ifdef TIMER
616                 functionTimer++;
617                 if(!(functionTimer%TIMERCOUNTER))
618                 {
619                         newTime = clock()-oldTime;
620                         double time=((float)(newTime)/CLOCKS_PER_SEC);
621 
622                         cout << "Timer: " << functionTimer << endl;
623                         cout << time/TIMERCOUNTER << " s pro Renderschritt" << endl;
```

```
624                         cout << TIMERCOUNTER/time << " Frames pro Sekunde" << endl;
625                         oldTime=clock();
626                 }
627         #endif
628
629 }
630
631
632
633 void Animate (int value)
634 {
635 // An dieser Stelle werden Berechnungen durchgef?hrt, die zu einer
636 // Animation der Szene erforderlich sind. Dieser Prozess l?uft im
637 // Hintergrund und wird hier alle 50 msec aufgerufen.
638
639 // RenderScene aufrufen
640 glutPostRedisplay();
641 // Timer wieder registrieren - Animate wird so wieder aufgerufen
642 glutTimerFunc(50,Animate,value);
643 }
644
645 void reshape( int w, int h )
646 {
647         // Prevent a divide by zero, when window is too short
648         // (you cant make a window of zero width).
649         if(h == 0)
650                 h = 1;
651
652         float ratio = 1.0f * w / h;
653         // Reset the coordinate system before modifying
654         glMatrixMode(GL_PROJECTION);
655         glLoadIdentity();
656
657         // Set the viewport to be the entire window
658     glViewport(0, 0, w, h);
659
660         // Set the clipping volume
661         gluPerspective(80,ratio,1,3000);
662         glMatrixMode(GL_MODELVIEW);
663         glLoadIdentity();
664 }
665
666
667 Navi3D_Display::Navi3D_Display(void)
668 {
669 }
670
671 Navi3D_Display::~Navi3D_Display(void)
672 {
673 }
674
675 void Navi3D_Display::changeArrowColor(int newColor)
676 {
677         pArrow->changeArrowColor(newColor);
678 }
679
680 void Navi3D_Display::setNewBaseFactor(float factor)
681 {
682         if(!(pArrow->setBaseFactor(factor)))
683                 printLog("Navi3D-Display.cpp : Error in setNewBaseFactor", 3);
684 }
685
686 void Navi3D_Display::ArrowSegmentation()
687 {
688         pArrow->changeArrowSegmentation();;
689 }
690
```

```
691 void Navi3D_Display::setNewArrowShadow(float newMaxShadow)
692 {
693         pArrow->setMaxShadow(newMaxShadow);
694 }
695
696 void Navi3D_Display::setNewArrowProfile(int profileNr)
697 {
698         if(!(pArrow->setNewArrowProfile(profileNr)))
699                 printLog("Navi3D-Display.cpp : Error in setNewArrowProfile", 3);
700 }
701
702 void Navi3D_Display::setArrowBegin(int ArrowBegin)
703 {
704         if(!(pArrow->setArrowBegin(ArrowBegin)))
705                 printLog("Navi3D-Display.cpp : Error in setArrowBegin", 3);
706 }
707
708 void Navi3D_Display::setArrowPeakType(int ArrowPeakType)
709 {
710         if(!(pArrow->setArrowPeakType(ArrowPeakType)))
711                 printLog("Navi3D-Display.cpp : Error in setArrowPeakType", 3);
712
713 }
714
715 void Navi3D_Display::calculateArrow()
716 {
717         if(!(pArrow->calculateArrowData()))
718                 printLog("Navi3D-Display.cpp : Error in calculateArrow", 3);
719 }
720
721 Navi3D_Display::Navi3D_Display(int argc, char** argv, IGADA* _pGADA, Navi3D_Base* _pBase)
722 {
723         //
724         Augenabstand=0.01f;
725         eyeX = 0;
726         eyeY = 4;
727         eyeZ = 9;
728
729         //* Set DEBUG pointers to manipulate interpolator during runtime
730
731
732         if(systemPar.KERNEL.interpolPoints.enable)
733                 {
734         glutGADA = static_cast<Navi3D_InterpolatePoints*>(_pGADA);
735         glutInterpolator = static_cast<KBInterpolator*>(glutGADA->getInterpolator());
736                 }
737                 //newValue = static_cast<Navi3D_InterpolatePoints*>(glutGADA)->getTension() + 0.1f;
738
739         //Create and set Pointers
740         pDisplay = this;
741         pGADA = _pGADA;
742         pBase = _pBase;
743         systemPar = pBase->getSystemParameters();
744         pSunpos = new Navi3D_SunPos();
745         pPOI = new Navi3D_POI(_pBase, _pGADA);
746
747         pGIMMICK = new Navi3D_GIMMICK(_pBase, _pGADA);
748         pGIMMICK_global = pGIMMICK;
749
750 #ifdef FAHR
751         pFahrspur = new Navi3D_Fahrspur();
752 #endif
753         pArrow = new Navi3D_Arrow(_pBase, _pGADA);
754
755
756         //Glut Functions
757         glutInit(&argc, argv); // Erm Just Write It =)
```

```
758         glutInitDisplayMode ( GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA ); // Display Mode
759         glutInitWindowPosition (0,0);
760         glutInitWindowSize  ( 500, 500 ); // If glutFullScreen wasn't called this is the window size
761         glutCreateWindow    ( "Navi3D" ); // Window Title (argv[0] for current directory as title)
762         init ();
763         glutFullScreen      (  );          // Put Into Full Screen
764
765 #ifdef TIMER
766         oldTime = clock();
767 #endif
768
769         glutDisplayFunc     ( display );  // Matching Earlier Functions To Their Counterparts
770         glutReshapeFunc     ( reshape );
771         glutKeyboardFunc    ( keyboard );
772         glutSpecialFunc     ( arrow_keys );
773         glutIdleFunc            ( display );
774
775         glutTimerFunc(50,Animate,0);
776
777         mouseMenu();
778
779         glutMainLoop        ( );          // Initialize The Main Loop*/
780 }
781 float demotime = 12.0; //Demotime for SunPos
782
783 void Navi3D_Display::init(void)
784 {
785         //activate Depth Test
786         glEnable(GL_DEPTH_TEST);
787         glDepthFunc(GL_LEQUAL);
788         glClearDepth(1.0f);
789
790     glClearColor(0.0f, 0.0f, 0.0f, 0.5f);                               // Black Background
791
792 if(systemPar.GADA.mode == BECKER_DEMO || systemPar.GADA.mode == STANDARD_ARROW)
793 {
794                 pSunpos->setDate("05.05.2009");
795                 pSunpos->setLatitude((float)49.94);
796                 pSunpos->setLongitude((float)8.74);
797                 pSunpos->setSunRadius(20);
798                 pSunpos->setTime(demotime);
799                 pSunpos->setTimezone(0.0);
800                 pSunpos->setHeading(180.0);
801 }
802 else
803 {
804                 pSunpos->setDate(pGADA->getMonth(),pGADA->getDay());
805                 pSunpos->setLatitude((float)pGADA->getLatitude()/1000.0);
806                 pSunpos->setLongitude((float)pGADA->getLongitude()/1000.0);
807                 pSunpos->setTime(pGADA->getUtc_hour(),pGADA->getUtc_minute());
808                 pSunpos->setTimezone(1.0);
809                 pSunpos->setSunRadius(10);
810                 pSunpos->setHeading(pGADA->getHeading());//ohne Target geht das wohl nicht!
811 }
812
813 #ifdef SHADER
814 {
815         glGenTextures(1, &depthTex);
816         glBindTexture(GL_TEXTURE_2D, depthTex);
817         glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT,512,512, 0,GL_DEPTH_COMPONENT,
              GL_UNSIGNED_BYTE, 0);
818         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
819         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
820         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
821         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
822         glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE, GL_COMPARE_R_TO_TEXTURE);
823
```

```
824         //Adressen der Multitexturing-Funktionen setzen
825         //glActiveTexture1 = (PFNGLACTIVETEXTUREPROC) glXGetProcAddress((const unsigned char*)"
                glActiveTexture");
826         //glMultiTexCoord2f1 = (PFNGLMULTITEXCOORD2FPROC) glXGetProcAddress((const unsigned char*)"
                glMultiTexCoord2f");
827         //glMultiTexCoord3f1 = (PFNGLMULTITEXCOORD3FPROC) glXGetProcAddress((const unsigned char*)"
                glMultiTexCoord3f");
828
829         //Texturen laden
830         CreateTGATexture(&texture[5], "Textures/Steinboden.tga");
831         CreateTGATexture(&texture[6], "Textures/Sonne.tga");
832
833         //glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
834         glEnable(GL_DEPTH_TEST);
835         glDepthFunc(GL_LEQUAL);
836         glClearDepth(1.0f);
837
838         //Shader initialisieren
839         bool glslSupported = InitGLSL();
840         if(!glslSupported)
841                 cout<<"Error Initializing GLSL"<<endl;
842         else
843                 cout<<"Success Initializing GLSL"<<endl;
844
845         //Shader laden
846         LoadShader(&shader_Pfeil, &VS_Pfeil, &PS_Pfeil, "Shader//FX//VS_Pfeil.txt", "Shader//FX//
                PS_Pfeil.txt");
847         LoadShader(&shader_Steinboden, &VS_Steinboden, &PS_Steinboden, "Shader//FX//VS_Steinboden.txt"
                , "Shader//FX//PS_Steinboden.txt");
848         LoadShader(&shader_Sonne, &VS_Sonne, &PS_Sonne, "Shader//FX//VS_Sonne.txt", "Shader//FX//
                PS_Sonne.txt");
849
850         //Lichtquellen Eigenschaften setzen
851         float lightambient[4] = {0.5,0.5,0.5,1.0};
852         glLightfv(GL_LIGHT0,GL_AMBIENT,lightambient);
853         float lightdiffuse[4] = {0.5,0.5,0.5,1.0};
854         glLightfv(GL_LIGHT0,GL_DIFFUSE,lightdiffuse);
855         float lightspecular[4] = {1.0,1.0,1.0,1.0};
856         glLightfv(GL_LIGHT0,GL_SPECULAR,lightspecular);
857
858         lightposition[0] = pSunpos->getSunPosX();
859         lightposition[1] = pSunpos->getSunPosY();
860         lightposition[2] = pSunpos->getSunPosZ();
861
862         glLightfv(GL_LIGHT0,GL_POSITION,lightposition);
863
864
865         }
866 #else
867         {
868
869         glEnable(GL_NORMALIZE);
870         glEnable(GL_LIGHTING); // Activate Light
871         glShadeModel(GL_SMOOTH); //Activate Shading
872         //set Lights
873         //glLightfv(GL_LIGHT1, GL_AMBIENT, LightAmbient);                    // Setup The
                Ambient Light
874         glLightfv(GL_LIGHT1, GL_DIFFUSE, LightDiffuse);                    // Setup The Diffuse
                Light
875         glLightfv(GL_LIGHT1, GL_POSITION,LightPosition);                   // Position The Light
876         glEnable(GL_LIGHT1);
877
878         glEnable(GL_COLOR_MATERIAL); //Enable the Color Material
879         }
880 #endif
881
882 #ifdef FOG
```

```
883         glFogfv(GL_FOG_COLOR, color);
884         glFogf(GL_FOG_START, FOG_START);
885         glFogf(GL_FOG_END, FOG_END);
886         glFogi(GL_FOG_MODE, GL_LINEAR);
887         glEnable(GL_FOG);
888 #endif
889
890 }
891
892
893
894 void Navi3D_Display::Geometry(){
895
896 if(systemPar.GADA.mode == BECKER_DEMO || systemPar.GADA.mode == STANDARD_ARROW)
897 {
898
899         /*Demo for Sunpos*/
900         if(timeON)
901                 demotime += (float)0.05;
902
903         if(demotime > 19)
904                 demotime = 5.0;
905
906         pSunpos->setTime(demotime);
907
908         /*demo for Heading*/
909         //demoHead += (float)1.0;
910         //
911         if(demoHead > 360.0)
912                 demoHead=0.0;
913
914         pSunpos->setHeading(demoHead);
915 }
916 else
917 {
918         pSunpos->setDate(pGADA->getMonth(),pGADA->getDay());
919         pSunpos->setLatitude((float)pGADA->getLatitude()/1000.0);
920         pSunpos->setLongitude((float)pGADA->getLongitude()/1000.0);
921         pSunpos->setTime(pGADA->getUtc_hour(),pGADA->getUtc_minute());
922         pSunpos->setTimezone(1.0);
923         pSunpos->setSunRadius(10);
924         pSunpos->setHeading(pGADA->getHeading());//ohne Target geht das wohl nicht!
925 }
926
927 #ifdef SHADER
928         ////////////////////////////////////////
929         // Shadowmap erzeugen
930         ////////////////////////////////////////
931         glViewport(0,0,512,512); //Setzt den Viewport auf 512x512 die größe der Shaowmap Textur
932         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //Leert die Buffer
933         glLoadIdentity();
934         gluLookAt(pSunpos->getSunPosX(), pSunpos->getSunPosY(), pSunpos->getSunPosZ(), 0.0015f,0.001f
                ,0.005f, 0,1,0); //setzt die Kamera auf Poistion der Lichtquelle
935
936         glGetFloatv(GL_MODELVIEW_MATRIX, mv);
937         glGetFloatv(GL_PROJECTION_MATRIX, proj);
938
939         //bauSchattenSzene(); //baut die Szene nach
940         pArrow->Arrow_Display();
941
942
943         glBindTexture(GL_TEXTURE_2D, depthTex); //Bindet die Textur
944         glCopyTexImage2D(GL_TEXTURE_2D, 0,GL_DEPTH_COMPONENT, 0,0,512,512,0); //Speichert das Bild in
                die depthTex Textur
945
946         lightposition[0] = pSunpos->getSunPosX();
947         lightposition[1] = pSunpos->getSunPosY();
```

```
948         lightposition[2] = pSunpos->getSunPosZ();
949 #endif
950         glPushMatrix();
951                 LightPosition[0] = pSunpos->getSunPosX();
952                 LightPosition[1] = pSunpos->getSunPosY();
953                 LightPosition[2] = pSunpos->getSunPosZ();
954         glPopMatrix();
955
956         glViewport (0, 0, winwidth,winheight); //setzt den Viewport wieder zurück
957         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //Leert die Buffer
958         glLoadIdentity();
959         gluLookAt(pDisplay->eyeX+pDisplay->Augenabstand, pDisplay->eyeY, pDisplay->eyeZ,
960                         0+pDisplay->Augenabstand, 0, -10,
961                             0.0f,1.0f,0.0f);
962 #ifdef SHADER
963         glUseProgramObjectARB(shader_Sonne);
964         glActiveTexture(GL_TEXTURE0); //erste textureinheit aktivieren...
965         glBindTexture(GL_TEXTURE_2D, texture[6]); //...und die in die textur gerenderte szene binden
966         Sun(LightPosition[0],LightPosition[1],LightPosition[2]);
967
968
969         glUseProgramObjectARB(shader_Steinboden); //Initialisiert den Shader für den Steinboden
970
971         shadowBegin();
972
973         int uniform = glGetUniformLocationARB(shader_Steinboden, "depthTex");
974         glUniform1iARB(uniform, 2); //erstes bild kommt in erste textureinheit
975
976         uniform = glGetUniformLocationARB(shader_Steinboden, "lightPos");
977         glUniform4fARB(uniform, pSunpos->getSunPosX(), pSunpos->getSunPosY(),  pSunpos->getSunPosZ(),
                1.0f);
978
979         glActiveTexture(GL_TEXTURE2);
980         glBindTexture(GL_TEXTURE_2D, depthTex); //...und die in die textur gerenderte szene binden
981
982         Ground();
983         shadowEnd();
984
985         glUseProgramObjectARB(shader_Pfeil);
986 #endif
987
988         pArrow->Arrow_Display();
989
990         glDisable(GL_LIGHT1);
991                 glLightfv(GL_LIGHT1, GL_POSITION,LightPosition);                        // Position
                        The Light
992         glEnable(GL_LIGHT1);
993
994         cout << pSunpos->getSunPosX() << "  " << pSunpos->getSunPosY() << "  "<< pSunpos->getSunPosZ()
                << "  " << endl;
995
996 #ifdef SHADER
997         glUseProgramObjectARB(0);
998 #endif
999
1000 #ifdef POI
1001         pPOI->DisplayPOI();
1002 #endif
1003
1004 #ifdef FAHR
1005         pFahrspur->detection();
1006 #endif
1007
1008         glPopMatrix();
1009
1010                 // Display sentence
1011         glDisable(GL_LIGHTING);
```

```
1012                    glPushMatrix();
1013                            glColor3f(1.0f,1.0f,1.0f);
1014                            setOrthographicProjection();
1015                            glLoadIdentity();
1016                            if(systemPar.KERNEL.arrowSpeedCollor.enable){
1017                                    renderSpacedBitmapString((winwidth/2)-150,winheight-100,10,(void *)
                                            font,pArrow->getNextCurveInMeter());
1018                            }
1019    #ifdef GIM
1020                            renderSpacedBitmapString((winwidth/2)-150,winheight-150,10,(void *)font,
                                    pGIMMICK->getSpeedInKMH());
1021                            if(pGIMMICK->getWarning())
1022                                    renderSpacedBitmapString((winwidth/2)-150,winheight-200,10,(void *)
                                            font,"slow down");
1023    #endif
1024
1025                            resetPerspectiveProjection();
1026                            glPopMatrix();
1027                    glPopMatrix();
1028            glEnable(GL_LIGHTING);
1029    }
```

A.4.13 Navi3D_Client: Headerdatei

```
1   ////////////////////////////////////////////////////////////////////////////////
2   //
3   //                      Hochschule Darmstadt - SS09 - Navi3D
4   //                              Copyright by h-da
5   //
6   //                      Created: Marco Muench / 2009/04/21
7   //
8   //                      Person in charge : ---
9   //
10  //                      Edit by: Marco Muench, 24.6.2009
11  //
12  //                      BaseClass of Navi3D-Application
13  //
14  ////////////////////////////////////////////////////////////////////////////////
15
16
17  #pragma once
18  #include "Define.h"
19
20
21  #include "ARROW/Navi3D_InterpolatePoints.h"
22  #include "KERNEL/Navi3D_Display.h"
23  #include "XML/Navi3D_Base.h"
24
25
26  #include <iostream>
27  #include <string>
28  #include <fstream>
29  #include <ctime>
30
31
32  using namespace std;
33
34  class Navi3D_Client
35  {
36  public:
37          Navi3D_Client(void);
38          ~Navi3D_Client(void);
39          Navi3D_Client(int argc, char** argv, unsigned int levelOfWriteLog );
40  private:
41          //Navi3D_InterpolatePoints *points;
42          Navi3D_Display *display;
43          Navi3D_Base *base;
```

```
44          GADA* gada;
45          N3D_System systemPar;
46          bool connected;
47  };
```

A.4.14 Navi3D_Client: Sourcedatei

```
1   #include "KERNEL/Navi3D_Client.h"
2   #include "ARROW/KBInterpolator.h"
3
4   unsigned int logLevel;
5
6   ////////////////////////////////////////
7   // Print Log
8   //
9   // Print a String into the Logfile
10  // Level = 1 : Very important Messages (Errors etc.)
11  // Level = 2 : Medium  Messages
12  // Level = 3 : Standard Messages
13  ////////////////////////////////////////
14  void printLog(string Log, unsigned int Level)
15  {
16          if(logLevel == 0 || Level > 3)
17                  return;
18
19          fstream save;   //open the File
20          save.open("log.txt", ios::out | ios::app);
21
22          if(Level <= logLevel )
23          {
24                  time_t t = time(0);
25                  string time =  ctime( &t );
26                  time.erase(time.length()-1,1); //delete the '\n' from ctime
27                  save << "[" << time << "] " << Log << endl;
28          }
29          save.close();
30  }
31
32  ////////////////////////////////////////
33  // delete Logfile
34  ////////////////////////////////////////
35  void deleteLog()
36  {
37          fstream save;
38          save.open("log.txt", ios::out | ios::trunc);
39          save << "==== NAVI 3D LOG ====" << endl;
40          time_t t = time(0);
41          string time =  ctime( &t );
42          save << " -> Start Time is " << time << endl;
43          save.close();
44  }
45
46  Navi3D_Client::Navi3D_Client(void)
47  {
48
49  }
50
51  Navi3D_Client::Navi3D_Client(int argc, char** argv, unsigned int levelOfWriteLog )
52  {
53
54
55  //      points = new Navi3D_InterpolatePoints();
56          //gada = new GADA("127.0.0.1");
57          //GADA*
58
59          logLevel = levelOfWriteLog;     //set Log Level
60          deleteLog();                            //delete the present Log File
```

```
61          base = new Navi3D_Base();
62          //load xml file
63          base->loadXML(XMLfile);
64
65          //get connection data from xml file
66          systemPar = base->getSystemParameters();
67
68          //load log level from xml file
69          logLevel = systemPar.KERNEL.logLevel;
70
71          //connection for beckerdemo and target are the same
72          if(systemPar.GADA.mode == BECKER_DEMO || systemPar.GADA.mode == TARGET)
73                  gada = new GADA(systemPar);
74          else
75                  gada = new GADA(systemPar);
76
77
78  if(systemPar.GADA.waitForConnect)
79  {
80          while(!gada->connect())
81          {
82                  cout << "please activate BeckerDemo or Target. " << endl;
83          }
84  }
85  else
86  {
87          gada->connect();
88  }
89
90  if(systemPar.KERNEL.interpolPoints.enable)
91  {
92          IGenericInterpolator* interpolator = new KBInterpolator();
93          IGADA* gada_interpolator = new Navi3D_InterpolatePoints(gada, interpolator, systemPar);
94          display = new Navi3D_Display(argc, argv, gada_interpolator, base);
95  }
96  else
97  {
98          display = new Navi3D_Display(argc, argv, gada, base);
99  }
100
101 }
102
103 Navi3D_Client::~Navi3D_Client(void)
104 {
105         delete gada;
106         delete base;
107         delete display;
108 }
```

A.4.15 Define: Headerdatei

```
1  ////////////////////////////////////////////////////////////////////////////////
2  //
3  //                      Hochschule Darmstadt - SS09 - Navi3D
4  //                              Copyright by h-da
5  //
6  //                      Created: Marco Muench / 2009/05/10
7  //
8  //                      Person in charge : ---
9  //
10 //                      Edit by: Marco Muench, 17.5.2009
11 //
12 //                      OpenGL Navi3D-Application
13 //
14 ////////////////////////////////////////////////////////////////////////////////
15
16 //Here you can define Defines ^^
```

```
#define XMLfile "Profiles.xml"
#define SHADER //activate Shader

//all done by xml file now
//#define BECKER_MODEa //activate the Becker DEMO Mode
//#define STANDARD_ARROWa //activate a Arrow with NO function and no movement
//#define WAIT_FOR_CONNECTaus //Wait for Connection with BeckerDemo or Target
//#define STEREOPROJECTION
//#define INTERPOLATE_POINTS //activate Interpolate Points
//#define FOG //acivate Fog
//#define RADIUSEDSTART //the begin of the arrow are Radiusd
//#define COLOR_ON_ARROW_SPEEDCONTROL //activate the

#define ROTATE //rotate the basepoints
#define TIMER
#define FAHRaus
#define GIM


///defines as parameter

#define TIMERCOUNTER 500
//#define GESCHWINDIGKEIT 100

//#define INTERPOLATION_RESOLUTION 10 // Defines in how many pieces a segment is splitted up
#define MAX_INTERPOLATION_RESOLUTION 50

//#define FOG_START 50
//#define FOG_END 100

//#define IP_ADDRESS "192.168.45.2"

//#define WINDOWS //LINUX
#define SQL_DATABASE
#define POIaus
```

A.4.16 DataObjects: Headerdatei

```
#pragma once

#include "Define.h"

typedef unsigned char UInt8;
typedef char Int8;
typedef unsigned short UInt16;
typedef short Int16;
typedef unsigned int UInt32;
typedef int Int32;
typedef float Float32;
typedef double Float64;
typedef char* String;


typedef struct
{
        UInt16 TagID; //sollte hier 'HO' sein
        UInt16 Length; //Headerlannge sollte hier 4 sein
        UInt16 size;    //lannge des folgenden streams in bytes;
        UInt16 objects; //lannge des folgenden streams in objekten;
}HeaderObject;

struct DataIndex
{
        UInt16 tag;                     //-> char[2] wie 'GA' f?r guidance arrow
        UInt16 offset;          // distance between overlayData pointer and data-beginning
        UInt16 length;          // length of tag-data
};
```

```
30
31  //for beckermode
32  struct GAHeader{
33          UInt16  tag;                    //tag id (should be GA)
34          UInt16  length;                 //GA length in bytes (after length-field)
35          UInt16  arrowAttributes;//bitfield: bit0 0=arrow on road surface, 1= floating arrow
36          UInt8   numberOfVertices;       //number of Vertices
37          Int16   manPointIndex;  // ManoeuvrePointIndex
38          UInt8   p1,p2,p3;               //padding bytes
39  };
40  //for target mode
41  struct GAHeaderT{
42          UInt16  tag;                    //tag id (should be GA)
43          UInt16  length;                 //GA length in bytes (after length-field)
44          UInt16  arrowAttributes;//bitfield: bit0 0=arrow on road surface, 1= floating arrow
45          UInt16  numberOfVertices;       //number of Vertices
46          Int16   manPointIndex;  // ManoeuvrePointIndex
47  };
48
49
50  struct GPSInfos{
51          Int32  longitude;       // current gps longitude
52          Int32  latitude;        // current gps latitude
53          UInt16 year;
54          UInt8  month;
55          UInt8  day;
56          UInt8  utc_hour;
57          UInt8  utc_minute;
58          UInt8  utc_second;
59          UInt32 speed;           // current speed
60          UInt32 heading;         // current heading
61          Int32  height;          // current gps height
62  };
63
64  struct GAVertex{
65          float x;
66          float y;
67          float z;
68          UInt32 Attributes;
69  };
70
71  struct GAdata{
72          GAHeader header;
73          GAHeaderT headerT;
74          GAVertex vertices[128];
75  };
76
77  struct GPSdata{
78          GAHeader header;
79          GAHeaderT headerT;
80          GPSInfos data;
81  };
82
83  inline GAVertex operator+ (const GAVertex &lhs, const GAVertex &rhs)
84  {
85          GAVertex out;
86          out.x = lhs.x + rhs.x;
87          out.y = lhs.y + rhs.y;
88          out.z = lhs.z + rhs.z;
89          return out;
90  }
91
92  inline GAVertex operator* (const GAVertex &lhs, const GAVertex &rhs)
93  {
94          GAVertex out;
95          out.x = lhs.x * rhs.x;
96          out.y = lhs.y * rhs.y;
```

```
        out.z = lhs.z * rhs.z;
        return out;
}

inline GAVertex operator* (const GAVertex &lhs, const float &rhs)
{
        GAVertex out;
        out.x = lhs.x * rhs;
        out.y = lhs.y * rhs;
        out.z = lhs.z * rhs;
        return out;
}
```

A.4.17 POI: Headerdatei

```
////////////////////////////////////////////////////////////////////////////////
//
//                      Hochschule Darmstadt - SS09 - Navi3D
//                              Copyright by h-da
//
//                      Created: Marco Muench / 2009/04/22
//
//                      Person in charge : Julia Runge
//                              Christian Bartel
//                              Antje Gloystein
//
//                      Edit by:
//
//                      POI
//
////////////////////////////////////////////////////////////////////////////////
#pragma once

#include <iostream>

#ifdef WINDOWS
#include <windows.h>
#include <winsock.h>
#endif

#include <stdio.h>
#include <stdlib.h>

#include <GL/glut.h>
#include <GL/gl.h>
#include <SDL/SDL.h>
#include <SDL/SDL_opengl.h>
#include <SDL/SDL_image.h>
#include "GADA/IGADA.h"
#include "K_SQLInterface.h"

#include "XML/Navi3D_Base.h"
#include "ARROW/Navi3D_InterpolatePoints.h"

using namespace std;

class Navi3D_POI{
public:
        Navi3D_POI(void); //Standardconstructor
        ~Navi3D_POI(void); //Standarddestructor
        Navi3D_POI(Navi3D_Base *tBase, IGADA *tGADA); //Constructor
        void DisplayPOI(); //display the POIs
        void loadTexture(char * name, GLuint *tex); //load Texture for a POI
        void readGPSData(); //read file with GPS-data
        K_SQLInterface SQLConnect(); //connect with interface
        void getPOI(); //get actual POIs
        void createCube(float x_local, float y_local, float z_local); //draw cube at given position
```

```
        Int32 longitudes[101]; //array for lognitude-values
        Int32 latitudes[101]; //array for latitude-values
        int ll_counter;
        int timer;

        GLuint texture; //actual texture
        K_SQLInterface sqlInterface; //connection with database
        structSimplePOISearch strSimplePOISearch; //struct for simple-search
        structComplexPOISearch strComplexPOISearch; //struct for complex search
        structPoint     strCarPosition; //actual carposition
        structPoint strBeginPoint; //beginpoint for POI-search
        structPoint strEndPoint; //endpoint for POI-search
        float x; //x-value of the POI-position
        float y; //y-value of the POI-position
        float z; //z-value of the POI-position
        float texProportion; //proportion from the texture
        float pseudo_z; //pseudo_z for simulated movement
        char filename[13]; //filename from texture
        vector<structPOI> * A;  //result from the static-circuit-search
        vector<structPOI> * B; //result from the flexible-circuit-search
        vector<structPOI> * C; //result from the corridor-search
        N3D_System sytemPar;
        int POIcount; //count of located POIs
        structPOI strPOI; //actual POI
        SDL_Surface *surface; //texture-surface
        int prevTex; //previous texture
        int POItype; //POI-type
        Navi3D_Base *pBase;

        IGADA *pGADA; //GADA_interface
        vector<N3D_BasePoints> vBasePointsArrow;
};
```

A.4.18 POI: Sourcedatei

```
#include "POI/Navi3D_POI.h"

//using namespace System;

//Standard-Constructor
Navi3D_POI::Navi3D_POI(void){}
//Standard-Destructor
Navi3D_POI::~Navi3D_POI(void){}

////////////////////////////////////////
// Constructor
////////////////////////////////////////
Navi3D_POI::Navi3D_POI(Navi3D_Base *tBase, IGADA *tGADA)
{
        pBase = tBase;
        pGADA = tGADA;

        sytemPar = pBase->getSystemParameters();

        POItype = 0;
        prevTex = 0;

        //sqlInterface = SQLConnect(); //Connection to database
        pseudo_z = 0; //Value for moving the POIs
        readGPSData(); // To read out GPS-Data from file only once at beginning of the program
        timer=0; // To get GPS-Data in a longer time period, not all values at once
        ll_counter=0; // Counter to walk through previously saved GPS-Data
}

//Read the GPS-data from a file
void Navi3D_POI::readGPSData() {
        GPSdata gpsData[2];
```

```
33          int counter = 0;
34          int lati_count = 0;
35          int longi_count = 0;
36
37          // Open file which was filled by GPS-Reciever
38          //FILE *fGpsData = fopen("test.txt","r+b");
39          FILE *fGpsData = fopen("gagps","r+b");
40
41
42          if( NULL == fGpsData ) {
43                  cout << "Kann \"gagps\" nicht oeffnen!" << endl;
44          }
45
46          // Just read out "real" data, to get Longitude and Latitude Data from Car
47          while(fread(&gpsData, sizeof(gpsData), 1, fGpsData) == 1) {
48                  int long0 = gpsData[1].data.longitude;
49                  int lati0 = gpsData[1].data.latitude;
50                  // Write GPS-Data into temporary Data-Structure
51                  if(long0 > 8600000 && long0 < 8800000)
52                  {
53                          longitudes[longi_count] = long0;
54                          longi_count++;
55                  }
56                  if(lati0 > 49000000 && lati0 < 51000000)
57                  {
58                          latitudes[lati_count] = lati0;
59                          lati_count++;
60                  }
61                  counter ++;
62          }
63          cout << "Anzahl Datensaetze gelesen: " << lati_count << endl << endl;
64          fclose(fGpsData);
65  }
66
67  //Connecting to the database
68  K_SQLInterface Navi3D_POI::SQLConnect()
69  {
70          //get connection data from xml file
71          N3D_System system = pBase->getSystemParameters();
72
73          structSQLConnectionData strSQLConnectionData;
74          strSQLConnectionData.sSQLUserName = system.SQL_DB.user; //User-name
75          strSQLConnectionData.sSQLHost = system.SQL_DB.IP; //Host
76          strSQLConnectionData.sSQLPasswort = system.SQL_DB.password; //Password
77          strSQLConnectionData.sSQLDatabase = system.SQL_DB.dbName; //Database
78          strSQLConnectionData.iSQLPortNr = system.SQL_DB.port; //Port
79
80          K_SQLInterface asd (strSQLConnectionData);
81          return asd;
82
83
84  }
85
86  //Find all relevant POIs
87  void Navi3D_POI::getPOI(){
88          bool becker = false;
89          int lati;
90          int longi;
91  //it is not possible to find the member variable of the own class at this point, wtf, resolved like
        this:
92  N3D_System systemPar = pBase->getSystemParameters();
93
94  if(systemPar.GADA.mode == BECKER_DEMO || systemPar.GADA.mode == STANDARD_ARROW)
95  {
96          becker = true;
97  }
98  else
```

```
99  {
100         becker = false;
101 }
102
103         if(timer % 1 == 0 )//&& ll_counter < 101)
104         {
105                 //If becker-mode is active the actual car-position ist provided from a file with gps-
                        data, if not the GADA-interface provides the actual carposition
106                 if(becker)
107                 {
108                         // Some problems with POI-Database, so we set default values for
109                         // longitude and latitude. In this case some POIs are always available.
110                         lati = 4987920;
111                         //latitudes[ll_counter];
112                         longi = 864378;
113                         //longitudes[ll_counter];
114                         ll_counter++;
115                 } else
116                 {
117                         lati = pGADA->getLatitude();
118                         longi = pGADA->getLongitude();
119                 }
120
121                 strCarPosition.iLatitude = lati;  // Carposition as latitude-value
122                 strCarPosition.iLongitude = longi; // Carposition as longitude-value
123                 strComplexPOISearch.dCarRotationDegree = 0; // Carrotation in degrees
124                 strComplexPOISearch.iPOIRadiusBack = 0; //Radius for POI-search at the back of the car
125                 strComplexPOISearch.iPOIRadiusSide = 2; //Radius for POI-search at the car-sides
126                 strComplexPOISearch.iPOIRadiusFront = 0; //Radius for POI-search at the front of the
                        car
127                 strComplexPOISearch.iPOIAngleFront = 45; //Viewing-angle at the front of the car
128                 strComplexPOISearch.iPOIAngleBack = 45; //Viewing-angle at the back of the car
129                 strComplexPOISearch.iCityRadius = 1000; // Radius for POI-search in the city
130                 strComplexPOISearch.strCarPosition = strCarPosition;
131
132                 strSimplePOISearch.iPOIRadius = 5000; // Radius for simplePOISearch
133                 strSimplePOISearch.iCityRadius = 10000; // Radius of the actual city for
                        SimplePOISearch
134                 strSimplePOISearch.strCarPosition = strCarPosition;
135
136                 strBeginPoint.iLatitude = strCarPosition.iLatitude; //lati;
137                 strBeginPoint.iLongitude = strCarPosition.iLongitude; //longi;
138
139                 strEndPoint.iLatitude = pGADA->getVertex(pGADA->verticesCount()-1).x;//strCarPosition.
                        iLatitude+vBasePointsArrow[vBasePointsArrow.size()-1].y; // Position des Autos
                        nach x Metern
140                 strEndPoint.iLongitude = pGADA->getVertex(pGADA->verticesCount()-1).y;
141
142
143                 try
144                 {
145
146                         A = sqlInterface.GetPOI(strSimplePOISearch);
147                         B = sqlInterface.GetPOI(strComplexPOISearch);
148                         C = sqlInterface.GetPOI(strSimplePOISearch,strBeginPoint,strEndPoint);
149
150                         A = sqlInterface.GetPOIStaticCircuit(strSimplePOISearch,A); //result from the
                                static-circuit-search
151                         B = sqlInterface.GetPOIFlexibleCircuit(strComplexPOISearch,B); //result from
                                the flexible-circuit-search
152                         C = sqlInterface.GetPOICorridor(strSimplePOISearch,strBeginPoint,strEndPoint,C
                                ); //result from the corridor-search
153
154
155                         //iterate through all POIs from static-circuit-search and draw them
156                         //for simulated movement comment '+pzeudo_z' in
157                         POIcount = (int)A->size();
```

```
158                         for (int i = 0; i < POIcount; i++)
159                         {
160                                 strPOI = A->back();
161                                 A->pop_back(); //Pop from stack
162                                 POItype = strPOI.iPOI_Typ; //save type
163                                 z= strCarPosition.iLatitude-strPOI.iLatitude; //calc x- and z-value
164                                 x= strCarPosition.iLongitude-strPOI.iLongitude;
165                                 y = 2; //set y-value
166                                 createCube(x, y, z/*+pseudo_z*/); //draw cube with actual texture and
                                        position
167                         }
168
169                         POIcount = B->size();
170                         for (int i = 0; i < POIcount; i++)
171                         {
172                                 strPOI = B->back();
173                                 B->pop_back(); //Pop from stack
174                                 POItype = strPOI.iPOI_Typ; //save type
175                                 z= strPOI.iLatitude-strCarPosition.iLatitude; //calc x- and z-value
176                                 x= strPOI.iLongitude-strCarPosition.iLongitude;
177                                 y = 2; //set y-value
178                                 createCube(x, y, z/*+pseudo_z*/); //draw cube with actual texture and
                                        position
179                         }
180
181                         POIcount = C->size();
182                         for (int i = 0; i < POIcount; i++)
183                         {
184                                 strPOI = C->back();
185                                 C->pop_back(); //Pop from stack
186                                 POItype = strPOI.iPOI_Typ; //save type
187                                 z= strPOI.iLatitude-strCarPosition.iLatitude; //calc x- and z-value
188                                 x= strPOI.iLongitude-strCarPosition.iLongitude;
189                                 y = 2; //set y-value
190                                 createCube(x, y, z/*+pseudo_z*/); //draw cube with actual texture and
                                        position
191                         }
192                 }
193                 catch(int iExeptCode)
194                 {
195                         //TODO: make new output window
196                         //Console::WriteLine(iExeptCode);
197                 }
198
199                 //catch(Exception ^e)
200                 catch(int e)
201                 {
202                         //TODO: make new output window
203                         //Console::WriteLine(e->ToString());
204                 }
205         }
206         timer++;
207 }
208
209 //Load the actual texture
210 void Navi3D_POI::loadTexture(char * name, GLuint *tex){
211         surface = IMG_Load(name); //Load surface
212         glGenTextures(1, tex); //Generate texture
213         glBindTexture(GL_TEXTURE_2D, *tex); //Bind generated texture as actual texture
214         glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
215         glTexParameterf(GL_TEXTURE_2D, GL_GENERATE_MIPMAP, GL_TRUE);
216         glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST_MIPMAP_LINEAR);
217         glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST_MIPMAP_LINEAR);
218
219         gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB, surface->w, surface->h, GL_RGB,GL_UNSIGNED_BYTE,
                surface->pixels); //build mipmap
220         if (surface) SDL_FreeSurface(surface); //free surface
```

```
221 }
222
223 //create a cube with the active texture
224 //the height of the cube is dependent on the texture-proportion
225 void Navi3D_POI::createCube(float x_local, float y_local, float z_local){
226         sprintf(filename, "img/%d.png", POItype);
227         //if the previuos texture is the same, don't load it again
228         if(prevTex != POItype){
229                 loadTexture(filename, &texture);
230                 prevTex = POItype;
231         }
232
233         //draw cube with texture
234         glPushMatrix();
235         glEnable(GL_TEXTURE_2D);
236         glBindTexture(GL_TEXTURE_2D, texture);
237
238         //Translate to position
239         glTranslatef(x_local, y_local, z_local);
240
241         glBegin(GL_QUADS);
242         glTexCoord2i( 0, 1);
243         glVertex3f(-0.5,-0.5,-0.5);
244         glTexCoord2i( 1, 1);
245         glVertex3f(0.5,-0.5,-0.5);
246         glTexCoord2i( 1, 0);
247         glVertex3f(0.5,0.5,-0.5);
248         glTexCoord2i( 0, 0);
249         glVertex3f(-0.5,0.5,-0.5);
250         glEnd();
251
252         glBegin(GL_QUADS);
253         glTexCoord2i( 0, 1);
254         glVertex3f(-0.5,-0.5,-0.5);
255         glTexCoord2i( 1, 1);
256         glVertex3f(-0.5,-0.5,0.5);
257         glTexCoord2i( 1, 0);
258         glVertex3f(-0.5,0.5,0.5);
259         glTexCoord2i( 0, 0);
260         glVertex3f(-0.5,0.5,-0.5);
261         glEnd();
262
263         glBegin(GL_QUADS);
264         glTexCoord2i( 0, 1);
265         glVertex3f(-0.5,-0.5,0.5);
266         glTexCoord2i( 1, 1);
267         glVertex3f(0.5,-0.5,0.5);
268         glTexCoord2i( 1, 0);
269         glVertex3f(0.5,0.5,0.5);
270         glTexCoord2i( 0, 0);
271         glVertex3f(-0.5,0.5,0.5);
272         glEnd();
273
274         glBegin(GL_QUADS);
275         glTexCoord2i( 1, 1);
276         glVertex3f(0.5,-0.5,-0.5);
277         glTexCoord2i( 1, 0);
278         glVertex3f(0.5,0.5,-0.5);
279         glTexCoord2i( 0, 0);
280         glVertex3f(0.5,0.5,0.5);
281         glTexCoord2i( 0, 1);
282         glVertex3f(0.5,-0.5,0.5);
283         glEnd();
284
285         //Oben
286         /*glBegin(GL_QUADS);
287         glTexCoord2i( 1, 1);
```

```
288         glVertex3f(0.5,0.5,-0.5);
289         glTexCoord2i( 1, 0);
290         glVertex3f(-0.5,0.5,-0.5);
291         glTexCoord2i( 0, 0);
292         glVertex3f(-0.5,0.5,0.5);
293         glTexCoord2i( 0, 1);
294         glVertex3f(0.5,0.5,0.5);
295         glEnd();*/
296
297         glPopMatrix();
298         glDisable(GL_TEXTURE_2D);
299 }
300
301 void Navi3D_POI::DisplayPOI()
302 {
303         glColor3f(1.0f,1.0f,1.0f);
304
305         pseudo_z += 1.0f;
306         getPOI();
307
308 }
```

A.4.19 Main

```
1  #include <iostream>
2  #include "KERNEL/Navi3D_Client.h"
3
4  using namespace std;
5
6  int main(int argc, char** argv )
7  {
8         Navi3D_Client NaviClient(argc, argv,3);
9         return 0;
10 }
```

A.5 XML

Im folgenden ist der erzeugte Quellcode für den XML-Parser gelistet. Die Quelldateien von Tiny-XML werden hier nicht gesondert aufgeführt, da diese unter [Tho10] zu finden sind. Der Quellcode wird hier wie er im Projekt vorhanden ist dargestellt. Aus diesem Grund sind die Kommentare auch in Englisch gehalten. Da im Projekt Englisch als Sprache für Kommentare fest gelegt wurde. Im Anschluss wird der gesamte Aufbau einer XML-Datei mit verschiedenen Pfeilprofilen und den Systemparametern exemplarisch dargestellt. Der genaue Aufbau wurde in [BBB+09] und die Erweiterung in Kapitel 2.5 beschrieben.

Tobias Braun

A.5.1 XML-Modul: Headerdatei

```
1  ////////////////////////////////////////////////////////////////////////////////
2  //
3  //                      Hochschule Darmstadt - WS09/10 - Navi3D
4  //                              Copyright by h-da
5  //
6  //                      Created: Marco Muench / 2009/04/21
7  //
8  //                      Person in charge : Tobias Braun
9  //
10 //                      Edit by: Tobias Braun
11 //
12 //                      Navi3D Base Class of Navi3D-Application
13 //                      Create Arrow-Bases from XML-Files
14 //
15 ////////////////////////////////////////////////////////////////////////////////
```

```
16
17  #pragma once
18
19  #include <vector>
20  #include <list>
21  #include "TINYXML/tinyxml.h"
22  #include <string>
23
24  using namespace std;
25
26  //struct for color handling
27  //values from 0.0 to 1.0
28  struct Colors
29  {
30          float r;
31          float g;
32          float b;
33  };
34
35  //possible base types
36  //quad: a base with 4 points
37  //circle: a circle, it is defined over two parameters,
38  //if both are the same, a circle is created, if they are unequal a ellipse is created
39  //misc: a base with a variable amount of points ( at least 3)
40  enum N3D_BaseTypes{
41          Quad,
42          Circle,
43          Misc
44  };
45
46  //struct for the base points, which are defined in the xml file,
47  //the z value is later used for the creation of the arrow
48  struct N3D_BasePoints{
49          float x;
50          float y;
51          float z; //has to be 0 for base points
52  };
53
54  //struct for the different bases
55  struct N3D_Base{
56          vector<N3D_BasePoints> pPoints;
57          N3D_BaseTypes Type;
58  };
59
60  //struct for the parameters of the pike of the arrow
61  struct N3D_PIKE{
62          int baseType;
63          int renderType;
64          Colors color;
65  };
66
67  //struct for the parameters of the tail of the arrow
68  struct N3D_TAIL{
69          int baseType;
70          Colors color;
71  };
72
73  //struct for a profile
74  struct N3D_Profiles{
75          N3D_PIKE pike;
76          N3D_TAIL tail;
77  };
78
79  //struct for DB parameters
80  struct N3D_SQL_DB{
81          string IP;
82          int port;
```

```
83          string user;
84          string password;
85          string dbName;
86  };
87
88  //struct for fog parameters
89  struct N3D_FOG{
90          bool enable;
91          int start;
92          int end;
93  };
94
95  //struct for interpolation
96  struct N3D_INTERPOLATE_POINTS{
97          bool enable;
98          int resolution;
99          int maxResolution;
100 };
101
102 //struct for speed control
103 struct N3D_COLOR_ON_ARROW_SPEEDCONTROL{
104         bool enable;
105         int testSpeed;
106 };
107
108 //struct for kernel parameters
109 struct N3D_KERNEL{
110         int logLevel;
111         bool shader;
112         N3D_FOG fog;
113         N3D_INTERPOLATE_POINTS interpolPoints;
114         N3D_COLOR_ON_ARROW_SPEEDCONTROL arrowSpeedCollor;
115         bool STEREOPROJECTION;
116 };
117
118 enum conModes{STANDARD_ARROW, BECKER_DEMO, TARGET};
119
120 //this could be renamed, if gada is replaced by message parsing, in this case there
121 //is no need for a IP adress any more - maybe for becker demo it is needed
122 struct N3D_GADA{
123         string IP;
124         int port;
125         conModes mode;
126         bool waitForConnect;
127 };
128
129 //system struct
130 struct N3D_System{
131         N3D_SQL_DB              SQL_DB;
132         N3D_GADA                GADA;
133         N3D_KERNEL              KERNEL;
134 };
135
136 class Navi3D_Base
137 {
138 public:
139         Navi3D_Base(void);
140         ~Navi3D_Base(void);
141         //int getBaseCount();
142
143         //function to get the desired base points for the tail of the arrow from a profil
144         //the points will be stored in *points, if the vector is not empty,
145         //it will be cleard
146         bool getBasePoints(vector<N3D_BasePoints> *points, int profil);
147
148         //function to get the desired base points for the pike of the arrow from a profil
149         //the points will be stored in *points, if the vector is not empty,
```

```
150         //it will be cleard
151         bool getPikePoints(vector<N3D_BasePoints> *points, int profil);
152 
153         //fuction to get the desired base color from a profil
154         inline bool getBaseColor(Colors *color, int profil)
155         {
156                 if(profil > mMaxProfiles || profil <= 0)
157                         return false;
158                 *color = mProfiles.at(profil-1).tail.color;
159                 return true;
160         };
161 
162         //fuction to get the desired base color from a profil
163         inline bool getPikeColor(Colors *color, int profil)
164         {
165                 if(profil > mMaxProfiles || profil <= 0)
166                         return false;
167                 *color = mProfiles.at(profil-1).pike.color;
168                 return true;
169         };
170 
171         //function to get the desired render type for the arrow pike from a profil
172         //there are severeal possibilities, how the pile could be created,
173         //which should be used is stored in the xml file
174         inline bool getPikeRenderType(int *renderType, int profil)
175         {
176                 if(profil > mMaxProfiles || profil <= 0)
177                         return false;
178                 *renderType = mProfiles.at(profil-1).pike.renderType;
179                 return true;
180         };
181 
182         //this function loads and parses the xml file, it is used in the constructor of this class
183         void loadXML(string filename);
184 private:
185         //vector for the possible bases
186         vector<N3D_Base> mBases;
187         //vector for the profiles
188         vector<N3D_Profiles> mProfiles;
189         //count the maximum of loaded bases
190         int mMaxBases;
191         //count the macimum of loaded profiles
192         int mMaxProfiles;
193         //System variables
194         N3D_System mSystem;
195         //this function calculates from two radius values the points for a circle or an ellipse
196         bool getCirclePoints(N3D_BasePoints points, vector<N3D_BasePoints> *returnPoints);
197 
198 public:
199         //this function returns the amount of loaded profiles
200         inline int getMaxProfiles(void)
201 
202         {
203                 return mMaxProfiles;
204         }
205         inline N3D_System getSystemParameters(void)
206         {
207                 return mSystem;
208         }
209 
210 protected:
211         void loadBases(TiXmlHandle hDoc);
212         void loadProfiles(TiXmlHandle hDoc);
213         void loadSystem(TiXmlHandle hDoc);
214 };
```

A.5.2 XML-Modul: Sourcedatei

```
1
2  #include "XML/Navi3D_Base.h"
3  #define _USE_MATH_DEFINES
4  #include <math.h>
5
6  extern void printLog(string Log, unsigned int Level);
7
8  Navi3D_Base::Navi3D_Base(void)
9  : mMaxBases(0)
10 , mMaxProfiles(0)
11 {
12         //load xml file and parse values
13         //loadXML(XMLfile);
14 }
15
16 Navi3D_Base::~Navi3D_Base(void)
17 {
18         //free mem
19         if(mBases.empty() == false)
20                 mBases.clear();
21
22         if(!mProfiles.empty() == false)
23                 mProfiles.clear();
24 }
25
26 bool Navi3D_Base::getBasePoints(vector<N3D_BasePoints> *points, int profil)
27 {
28         //if vector is not empty, it will be cleared
29         if(points->empty() == false)
30                 points->clear();
31
32         //check if the desired profile number is valid
33         if(profil>mMaxProfiles || profil <= 0)
34                 return false;
35
36         //get the tail entry form the profil vector
37         int tail = mProfiles.at(profil-1).tail.baseType;
38
39         //check if the returned base number is valid
40         if(tail > mMaxBases)
41                 return false;
42
43         if(mBases.at(tail-1).Type == Quad || mBases.at(tail-1).Type == Misc)
44         {
45                 //get the points from the base vector
46                 *points = mBases.at(tail-1).pPoints;
47         }
48         else if(mBases.at(tail-1).Type == Circle)
49         {
50                 //calcs from the radius a circle and interpolates the points
51                 getCirclePoints(mBases.at(tail-1).pPoints.at(0), points);
52         }
53         else
54                 return false;
55
56         return true;
57 }
58
59 bool Navi3D_Base::getPikePoints(vector<N3D_BasePoints> *points, int profil)
60 {
61         //if vector is not empty, it will be cleared
62         if(points->empty() == false)
63                 points->clear();
64
65         //check if the desired profile number is valid
```

```
66          if(profil>mMaxProfiles || profil <= 0)
67                  return false;
68
69          //get the tail entry form the profil vector
70          int pike = mProfiles.at(profil-1).pike.baseType;
71
72          //check if the returned base number is valid
73          if(pike > mMaxBases)
74                  return false;
75
76          if(mBases.at(pike-1).Type == Quad || mBases.at(pike-1).Type == Misc)
77          {
78                  //get the points from the base vector
79                  *points = mBases.at(pike-1).pPoints;
80          }
81          else if(mBases.at(pike-1).Type == Circle)
82          {
83                  //calcs from the radius a circle and interpolates the points
84                  getCirclePoints(mBases.at(pike-1).pPoints.at(0), points);
85          }
86          else
87                  return false;
88
89          return true;
90  }
91
92  //load bases
93  void Navi3D_Base::loadBases(TiXmlHandle hDoc)
94  {
95          string name;
96          TiXmlElement* pElem;
97          TiXmlElement* pBase;
98          TiXmlElement* pPoints;
99
100         pElem=hDoc.FirstChild("Data").FirstChild("Bases").FirstChild("Base").ToElement();
101
102         // should always have a valid root but handle gracefully if it does
103         for(pElem; pElem; pElem=pElem->NextSiblingElement() )
104         {
105                 int num;
106                 if(pElem->QueryIntAttribute("num", &num)==TIXML_SUCCESS)
107                 {
108                         mMaxBases++;
109                         if(mMaxBases!=num)
110                         {
111                                 mMaxBases = 0;
112                                 printLog("Navi3D_Base.cpp: Error in XML-File, number of bases dosn't
                                        match",1);
113                                 return; //return if bases count not matches bases num from xml file
114                         }
115                         pBase = pElem->FirstChildElement();
116                         name=pBase->Value();
117
118                         if(name == "Quad")
119                         {
120                                 N3D_BasePoints points;
121                                 N3D_Base base;
122                                 base.Type = Quad;
123                                 pPoints = pBase->FirstChildElement("x");
124                                 if(pPoints)
125                                 {
126                                         //read the points (x and y) z is 0
127                                         for(pPoints; pPoints; pPoints=pPoints->NextSiblingElement("x")
                                                )
128                                         {
129                                                 points.x = (float)atof(  pPoints->GetText() );
```

```
130                                     points.y = (float)atof(  pPoints->NextSiblingElement("
                                            y")->GetText() );
131                                     points.z = 0.0f;
132                                     base.pPoints.push_back(points);
133                             }
134                             printLog("Navi3D_Base.cpp: Quad base found",3);
135                             mBases.push_back(base);
136                     }
137                     else
138                     {
139                             mMaxBases = 0;
140                             printLog("Navi3D_Base.cpp: Error in XML-File - Quad loading"
                                    ,1);
141                             return;
142                     }
143             }
144             else if(name == "Circle")
145             {
146                     N3D_BasePoints points;
147                     N3D_Base base;
148                     base.Type = Circle;
149                     pPoints = pBase->FirstChildElement("x");
150                     if(pPoints)
151                     {
152                             //read the points (x and y) z is 0
153                             for(pPoints; pPoints; pPoints=pPoints->NextSiblingElement("x")
                                    )
154                             {
155                                     points.x = (float)atof(  pPoints->GetText() );
156                                     points.y = (float)atof(  pPoints->NextSiblingElement("
                                            y")->GetText() );
157                                     points.z = 0.0f;
158                                     base.pPoints.push_back(points);
159                             }
160                             printLog("Navi3D_Base.cpp: Circle base found",3);
161                             mBases.push_back(base);
162                     }
163                     else
164                     {
165                             mMaxBases = 0;
166                             printLog("Navi3D_Base.cpp: Error in XML-File - Circle loading"
                                    ,1);
167                             return;
168                     }
169             }
170             else if(name == "Misc")
171             {
172                     N3D_BasePoints points;
173                     N3D_Base base;
174                     base.Type = Misc;
175                     pPoints = pBase->FirstChildElement("x");
176
177                     if(pPoints)
178                     {
179                             //read the points (x and y) z is 0
180                             for(pPoints; pPoints; pPoints=pPoints->NextSiblingElement("x")
                                    )
181                             {
182                                     points.x = (float)atof(  pPoints->GetText() );
183                                     points.y = (float)atof(  pPoints->NextSiblingElement("
                                            y")->GetText() );
184                                     points.z = 0.0f;
185                                     base.pPoints.push_back(points);
186                             }
187                             printLog("Navi3D_Base.cpp: Misc base found",3);
188                             mBases.push_back(base);
189                     }
```

```
190                                 else
191                                 {
192                                         mMaxBases = 0;
193                                         printLog("Navi3D_Base.cpp: Error in XML-File - Misc loading"
                                                ,1);
194                                         return;
195                                 }
196                         }
197                         else
198                         {
199                                 //error log undefined base
200                                 printLog("Navi3D_Base.cpp: Undefined base found",1);
201                         }
202                 }
203         }
204 }
205 //load profiles
206 void Navi3D_Base::loadProfiles(TiXmlHandle hDoc)
207 {
208         string name;
209         TiXmlElement* pElem;
210         TiXmlElement* pProfile;
211
212         pElem=hDoc.FirstChild("Data").FirstChild("Profiles").FirstChild("Profil").ToElement();
213
214         for(pElem; pElem; pElem=pElem->NextSiblingElement() )
215         {
216                 int num;
217                 if(pElem->QueryIntAttribute("num", &num)==TIXML_SUCCESS)
218                 {
219                         mMaxProfiles++;
220                         if(mMaxProfiles!=num)
221                         {
222                                 mMaxProfiles = 0;
223                                 printLog("Navi3D_Base.cpp: Error in XML-File, number of profiles dosn'
                                        t match",1);
224                                 return;
225                         }
226
227                         N3D_Profiles profile;
228                         //block: pike
229                         pProfile = pElem->FirstChildElement("pike")->FirstChildElement("basetype");
230                         if(pProfile)
231                         {
232                                 //basetype
233                                 profile.pike.baseType = atoi( pProfile->GetText() );
234                                 //rendertype
235                                 profile.pike.renderType = atoi(pProfile->NextSiblingElement("
                                        rendertype")->GetText());
236                                 //color
237                                 pProfile = pProfile->NextSiblingElement("color")->FirstChildElement("r
                                        ");
238                                 profile.pike.color.r = (float)atof( pProfile->GetText() );
239                                 profile.pike.color.g = (float)atof( pProfile->NextSiblingElement("g")
                                        ->GetText() );
240                                 profile.pike.color.b = (float)atof( pProfile->NextSiblingElement("b")
                                        ->GetText() );
241                         }
242                         else
243                         {
244                                 mMaxProfiles = 0;
245                                 printLog("Navi3D_Base.cpp: Error in XML-File - Pike loading",1);
246                                 return;
247                         }
248
249                         //block: tail
250                         pProfile = pElem->FirstChildElement("tail")->FirstChildElement("basetype");
```

```
251                         if(pProfile)
252                         {
253                                 //basetype
254                                 profile.tail.baseType = atoi(  pProfile->GetText() );
255                                 //color
256                                 pProfile = pProfile->NextSiblingElement("color")->FirstChildElement("r
                                        ");
257                                 profile.tail.color.r = (float)atof( pProfile->GetText() );
258                                 profile.tail.color.g = (float)atof( pProfile->NextSiblingElement("g")
                                        ->GetText() );
259                                 profile.tail.color.b = (float)atof( pProfile->NextSiblingElement("b")
                                        ->GetText() );
260                         }
261                         else
262                         {
263                                 mMaxProfiles = 0;
264                                 printLog("Navi3D_Base.cpp: Error in XML-File - Tail loading",1);
265                                 return;
266                         }
267
268                         if(profile.pike.baseType > mMaxBases || profile.tail.baseType > mMaxBases)
269                         {
270                                 mMaxProfiles = 0;
271                                 printLog("Navi3D_Base.cpp: Error in XML-File, number of profiles dosn'
                                        t match",1);
272                                 return;
273                         }
274                         mProfiles.push_back(profile);
275                 }
276         }
277 }
278
279 //load profiles
280 void Navi3D_Base::loadSystem(TiXmlHandle hDoc)
281 {
282         string name;
283         TiXmlElement* pElem;
284         TiXmlElement* pSystem;
285
286         pElem=hDoc.FirstChild("Data").FirstChild("System").ToElement();
287
288         for(pElem; pElem; pElem=pElem->NextSiblingElement() )
289         {
290                 //block: SQL_DB
291                 pSystem = pElem->FirstChildElement("SQL_DB")->FirstChildElement("IP");
292                 if(pSystem)
293                 {
294                         //IP
295                         mSystem.SQL_DB.IP = pSystem->GetText();
296                         //port
297                         mSystem.SQL_DB.port = atoi(pSystem->NextSiblingElement("port")->GetText());
298                         //user
299                         mSystem.SQL_DB.user = pSystem->NextSiblingElement("user")->GetText();
300                         //password
301                         mSystem.SQL_DB.password = pSystem->NextSiblingElement("password")->GetText();
302                         //db name
303                         mSystem.SQL_DB.dbName = pSystem->NextSiblingElement("db_name")->GetText();
304
305                         printLog("Navi3D_Base.cpp: SQL_DB settings found",3);
306                 }
307                 else
308                 {
309                         printLog("Navi3D_Base.cpp: Error in XML-File - System SQL_DB loading",1);
310                         return;
311                 }
312                 //block: GADA
313                 pSystem = pElem->FirstChildElement("GADA")->FirstChildElement("IP");
```

```
314                 if(pSystem)
315                 {
316                         //IP
317                         mSystem.GADA.IP = pSystem->GetText();
318                         //port
319                         mSystem.GADA.port = atoi(pSystem->NextSiblingElement("port")->GetText());
320                         mSystem.GADA.mode = (conModes) atoi(pSystem->NextSiblingElement("mode")->
                                GetText());
321                         mSystem.GADA.waitForConnect = atoi(pSystem->NextSiblingElement("wait")->
                                GetText());
322                         printLog("Navi3D_Base.cpp: GADA settings found",3);
323                 }
324                 else
325                 {
326                         printLog("Navi3D_Base.cpp: Error in XML-File - System GADA loading",1);
327                         //return;
328                 }
329
330                 //block: KERNEL
331                 pSystem = pElem->FirstChildElement("KERNEL")->FirstChildElement("loglevel");
332                 if(pSystem)
333                 {
334                         //IP
335                         mSystem.KERNEL.logLevel = atoi(pSystem->GetText());
336                         //port
337                         mSystem.KERNEL.shader = atoi(pSystem->NextSiblingElement("shader")->GetText())
                                ;
338                         mSystem.KERNEL.fog.enable = atoi(pSystem->NextSiblingElement("FOG")->
                                FirstChildElement("ON")->GetText());
339                         mSystem.KERNEL.fog.start = atoi(pSystem->NextSiblingElement("FOG")->
                                FirstChildElement("FOG_START")->GetText());
340                         mSystem.KERNEL.fog.end = atoi(pSystem->NextSiblingElement("FOG")->
                                FirstChildElement("FOG_END")->GetText());
341                         mSystem.KERNEL.interpolPoints.enable = atoi(pSystem->NextSiblingElement("
                                INTERPOLATE_POINTS")->FirstChildElement("ON")->GetText());
342                         mSystem.KERNEL.interpolPoints.resolution = atoi(pSystem->NextSiblingElement("
                                INTERPOLATE_POINTS")->FirstChildElement("INTERPOLATION_RESOLUTION")->
                                GetText());
343                         mSystem.KERNEL.interpolPoints.maxResolution = atoi(pSystem->NextSiblingElement
                                ("INTERPOLATE_POINTS")->FirstChildElement("MAX_INTERPOLATION_RESOLUTION")
                                ->GetText());
344                         mSystem.KERNEL.arrowSpeedCollor.enable = atoi(pSystem->NextSiblingElement("
                                COLOR_ON_ARROW_SPEEDCONTROL")->FirstChildElement("ON")->GetText());
345                         mSystem.KERNEL.arrowSpeedCollor.testSpeed = atoi(pSystem->NextSiblingElement("
                                COLOR_ON_ARROW_SPEEDCONTROL")->FirstChildElement("TEST_SPEED")->GetText()
                                );
346                         mSystem.KERNEL.STEREOPROJECTION = atoi(pSystem->NextSiblingElement("
                                STEREOPROJECTION")->GetText());
347                         printLog("Navi3D_Base.cpp: KERNEL settings found",3);
348                 }
349                 else
350                 {
351                         printLog("Navi3D_Base.cpp: Error in XML-File - System GADA loading",1);
352                         //return;
353                 }
354         }
355 }
356
357 void Navi3D_Base::loadXML(string filename)
358 {
359         TiXmlDocument doc(filename);
360         bool loadOkay = doc.LoadFile();
361         char txt[255];
362         string name;
363
364         if (loadOkay)
365         {
```

```
366                 TiXmlHandle hDoc(&doc);
367                 TiXmlHandle hRoot(0);
368                 sprintf(txt,"Navi3D_Base.cpp: File \"%s\" loaded successful", filename.c_str());
369                 printLog(txt,3);
370                 // block: bases
371                 loadBases(hDoc);
372                 //block: profiles
373                 loadProfiles(hDoc);
374                 //block: system
375                 loadSystem(hDoc);
376         }
377         else
378         {
379                 sprintf(txt,"Navi3D_Base.cpp: Failed to load file \"%s\"", filename.c_str());
380                 printLog(txt,1);
381         }
382 }
383
384 // calcs from radius a circle
385 bool Navi3D_Base::getCirclePoints(N3D_BasePoints points, vector<N3D_BasePoints> *returnPoints)
386 {
387         float radA = points.x;
388         float radB = points.y;
389         float gap = 20;
390         float maxSteps = 360;
391         float pi = (float) M_PI;
392         N3D_BasePoints tempPoints;
393         for(float i = maxSteps-10; i >= 0; i-=gap)
394         {
395                 tempPoints.x = sin((float)(i*pi/180.0f))*radA;
396                 tempPoints.y = cos((float)(i*pi/180.0f))*radB;
397                 tempPoints.z = 0.0f;
398                 returnPoints->push_back(tempPoints);
399         }
400         return true;
401 }
```

A.5.3 XML-Datei

```
1  <?xml version="1.0" standalone=no>
2
3  <Data>
4         <!-- Bases fuer the head and the tails of an arrow -->
5         <Bases>
6                 <Base num="1">
7                         <Quad>
8                                 <x>0.5</x> <y>0.5</y>
9
10                                <x>0.5</x> <y>-0.5</y>
11
12                                <x>-0.5</x> <y>-0.5</y>
13
14                                <x>-0.5</x> <y>0.5</y>
15
16                        </Quad>
17                </Base>
18                <Base num="2">
19                        <Quad>
20                                <x>0.5</x> <y>0.25</y>
21
22                                <x>-0.5</x> <y>0.25</y>
23
24                                <x>-0.5</x> <y>-0.25</y>
25
26                                <x>0.5</x> <y>-0.25</y>
27                        </Quad>
28                </Base>
```

```
29              <Base num="3">
30                      <Circle>
31                              <x>0.75</x> <y>0.75</y>
32                      </Circle>
33              </Base>
34              <Base num="4">
35                      <Quad>
36                              <x>2</x> <y>0.2</y>
37                              <x>2</x> <y>-0.2</y>
38                              <x>-2</x> <y>-0.2</y>
39                              <x>-2</x> <y>0.2</y>
40                      </Quad>
41              </Base>
42              <Base num="5">
43                      <Misc>
44                              <x>0.5</x> <y>0.5</y>
45
46                              <x>0.75</x> <y>0.25</y>
47
48                              <x>0.75</x> <y>-0.25</y>
49
50                              <x>0.5</x> <y>-0.5</y>
51
52                              <x>-0.5</x> <y>-0.5</y>
53
54                              <x>-0.75</x> <y>-0.25</y>
55
56                              <x>-0.75</x> <y>0.25</y>
57
58                              <x>-0.5</x> <y>0.5</y>
59                      </Misc>
60              </Base>
61              <Base num="6">
62                      <Misc>
63                              <x>0.0</x> <y>-0.25</y>
64
65                              <x>0.25</x> <y>0.25</y>
66
67                              <x>-0.25</x> <y>0.25</y>
68                      </Misc>
69              </Base>
70              <Base num="7">
71                      <Circle>
72                              <x>0.5</x> <y>0.25</y>
73                      </Circle>
74              </Base>
75      </Bases>
76
77      <!-- Two bases together form an arrow -->
78      <Profiles>
79              <Profil num="1">
80                      <pike>
81                              <basetype> 1 </basetype>
82                              <rendertype> 1 </rendertype>
83                              <color>
84                                      <r>1.0</r>
85                                      <g>0.0</g>
86                                      <b>0.0</b>
87                              </color>
88                      </pike>
89                      <tail>
90                              <basetype> 1 </basetype>
91                              <color>
92                                      <r>1.0</r>
93                                      <g>0.0</g>
94                                      <b>1.0</b>
95                              </color>
```

```
                </tail>
        </Profil>

        <Profil num="2">
                <pike>
                        <basetype> 2 </basetype>
                        <rendertype> 1 </rendertype>
                        <color>
                                <r>1.0</r>
                                <g>0.0</g>
                                <b>0.0</b>
                        </color>
                </pike>
                <tail>
                        <basetype> 2 </basetype>
                        <color>
                                <r>1.0</r>
                                <g>0.0</g>
                                <b>0.0</b>
                        </color>
                 </tail>
        </Profil>

        <Profil num="3">
                <pike>
                        <basetype> 2 </basetype>
                        <rendertype> 1 </rendertype>
                        <color>
                                <r>0.0</r>
                                <g>0.0</g>
                                <b>1.0</b>
                        </color>
                </pike>
                <tail>
                        <basetype> 5 </basetype>
                        <color>
                                <r>0.0</r>
                                <g>0.0</g>
                                <b>1.0</b>
                        </color>
                 </tail>
        </Profil>

        <Profil num="4">
                <pike>
                        <basetype> 2 </basetype>
                        <rendertype> 1 </rendertype>
                        <color>
                                <r>0.0</r>
                                <g>0.0</g>
                                <b>1.0</b>
                        </color>
                </pike>
                <tail>
                        <basetype> 3 </basetype>
                        <color>
                                <r>0.0</r>
                                <g>0.0</g>
                                <b>1.0</b>
                        </color>
                 </tail>
        </Profil>
        <Profil num="5">
                <pike>
                        <basetype> 1 </basetype>
                        <rendertype> 1 </rendertype>
                        <color>
```

```
                                        <r>0.0</r>
                                        <g>0.0</g>
                                        <b>1.0</b>
                                </color>
                        </pike>
                        <tail>
                                <basetype> 4 </basetype>
                                <color>
                                        <r>0.0</r>
                                        <g>0.0</g>
                                        <b>1.0</b>
                                </color>
                         </tail>
                </Profil>

        </Profiles>

        <System>
                <SQL_DB>
                        <IP>    127.0.0.1       </IP>
                        <port>  0       </port>
                        <user>  root    </user>
                        <password>      123     </password>
                        <db_name>       poi_database_germany    </db_name>
                </SQL_DB>

                <GADA>
                        <IP>    192.168.178.20  </IP>
                        <port>  7000    </port>

                        <!-- STANDARD_ARROW      0 -->
                        <!-- BECKER_DEMO                 1 -->
                        <!-- TARGET                              2 -->

                        <mode>  0</mode>

                        <!-- no 0       , yes 1 -->
                        <wait>  0       </wait>

                </GADA>

                <KERNEL>
                        <loglevel>      3       </loglevel>

                        <-- not used now , see for define in Define.h-->
                        <shader>        1       </shader>

                        <FOG>
                        <!-- no 0       , yes 1 -->
                                <ON>    1       </ON>
                                <FOG_START>     50      </FOG_START>
                                <FOG_END>       100     </FOG_END>
                        </FOG>
                        <INTERPOLATE_POINTS>
                                        <!-- no 0       , yes 1 -->
                                        <ON>    1       </ON>
                                        <INTERPOLATION_RESOLUTION>      10      </
                                            INTERPOLATION_RESOLUTION>

                                        <-- not used now , see for define in Define.h-->
                                        <MAX_INTERPOLATION_RESOLUTION>  50      </
                                            MAX_INTERPOLATION_RESOLUTION>

                        </INTERPOLATE_POINTS>

                        <COLOR_ON_ARROW_SPEEDCONTROL>
                                        <!-- no 0       , yes 1 -->
```

```
                                        <ON>    1       </ON>
                                        <!-- in demo mode there is no real speed, so this will
                                             be taken -->
                                        <TEST_SPEED>    100     </TEST_SPEED>
                        </COLOR_ON_ARROW_SPEEDCONTROL>

                        <!-- no 0       , yes 1 -->
                        <STEREOPROJECTION>      0       </STEREOPROJECTION>
                </KERNEL>
        </System>
</Data>
```

A.6 Sonnenstand und Shader

A.6.1 class Navi3D_SunPos.h

```
////////////////////////////////////////////////////////////////////////////////
//
//                      Hochschule Darmstadt - SS09 - Navi3D
//                              Copyright by h-da
//
//                      Created: Marco Münch / 2009/04/22
//
//                      Person in charge : Andreas Brust
//
//                      Edit by:
//
//                      Calculate the Sun-position
//
////////////////////////////////////////////////////////////////////////////////

#pragma once
#include <string>
using namespace std;

class Navi3D_SunPos
{
public:
        Navi3D_SunPos(void);
        ~Navi3D_SunPos(void);

        float getSunPosX(void);
        float getSunPosY(void);
        float getSunPosZ(void);

        float getSunHightAngle(void);
        float getSunAngleToNorth(void);

        bool setLatitude(float latitude);
        bool setLongitude(float longitude);
        bool setTimezone(float timezone);
        bool setDate(string date); //Bsp:"07.02.1984"
        bool setDate(int month, int day);
        bool setTime(string time); //Bsp:"06:24"
        bool setTime(int hour, int minute);
        bool setTime(float time);
        bool setSunRadius(float radius);
        bool setHeading(float angle);

private:
        bool calculateSunPos(void);

        float sunPos_x;
        float sunPos_y;
        float sunPos_z;

        float Latitude;
```

```
52          float Longitude;
53          float Timezone;
54          float c;
55          int Month;
56          int Day;
57          float Time;
58          float OldTime;
59          float Heading;
60
61          float Sun_HightAngle;
62          float Sun_AngleToNorth;
63
64          float Rad_Sun_AngleToNorth;
65          float Rad_Sun_HightAngle;
66          float Rad_Heading;
67
68          float Sun_Radius;
69  };
```

A.6.2 class Navi3D_SunPos.cpp

```
1   #include "Navi3D_SunPos.h"
2   #define _USE_MATH_DEFINES
3   #include <math.h>
4
5   Navi3D_SunPos::Navi3D_SunPos(void)
6   {
7           sunPos_x = 0.0;
8           sunPos_y = 0.0;
9           sunPos_z = 0.0;
10
11          Latitude = 0.0;
12          Longitude = 0.0;
13          Timezone = 1.0;
14          Month = 0;
15          Day = 0;
16          Time = 0.0;
17          OldTime = 0.0;
18
19          Sun_Radius = 5.0;
20          Sun_HightAngle = 0.0;
21          Sun_AngleToNorth = 0.0;
22
23          Rad_Sun_AngleToNorth = 0.0;
24          Rad_Sun_HightAngle = 0.0;
25          Rad_Heading = 0.0;
26  }
27
28  Navi3D_SunPos::~Navi3D_SunPos(void)
29  {
30  }
31
32  float Navi3D_SunPos::getSunPosX()
33  {
34          calculateSunPos();
35          return sunPos_x;
36  }
37
38  float Navi3D_SunPos::getSunPosY()
39  {
40          calculateSunPos();
41          return sunPos_y;
42  }
43
44  float Navi3D_SunPos::getSunPosZ()
45  {
46          calculateSunPos();
```

```
47          if(Sun_HightAngle < 0)
48                  return 0.0;
49          else
50                  return sunPos_z;
51  }
52
53  float Navi3D_SunPos::getSunAngleToNorth()
54  {
55          calculateSunPos();
56          return Sun_AngleToNorth;
57  }
58
59  float Navi3D_SunPos::getSunHightAngle()
60  {
61          calculateSunPos();
62          if(Sun_HightAngle < 0)
63                  return 0.0;
64          else
65                  return Sun_HightAngle;
66  }
67
68  bool Navi3D_SunPos::setLatitude(float latitude)
69  {
70          Latitude = latitude;
71          return true;
72  }
73
74  bool Navi3D_SunPos::setLongitude(float longitude)
75  {
76          Longitude = longitude;
77          return true;
78  }
79
80  bool Navi3D_SunPos::setTimezone(float timezone)
81  {
82          Timezone = timezone;
83
84          c = timezone * 15;
85
86          return true;
87  }
88
89  bool Navi3D_SunPos::setDate(string date)
90  {
91          int buffMonth = 0;
92
93          Day = atoi(date.substr(0,date.find(".")).c_str());
94          date = date.erase(0,date.find(".")+1);
95
96          buffMonth = atoi(date.substr(0,date.find(".")).c_str());
97          //date = date.erase(0,date.find(".")+1);
98
99          switch(buffMonth)
100         {
101                 case 1: Month = 0; break;
102                 case 2: Month = 31; break;
103                 case 3: Month = 59; break;
104                 case 4: Month = 90; break;
105                 case 5: Month = 120; break;
106                 case 6: Month = 151; break;
107                 case 7: Month = 181; break;
108                 case 8: Month = 212; break;
109                 case 9: Month = 243; break;
110                 case 10: Month = 273; break;
111                 case 11: Month = 304; break;
112                 case 12: Month = 334; break;
113         }
```

```
114         return true;
115 }
116
117 bool Navi3D_SunPos::setDate(int month, int day)
118 {
119         Day = day;
120
121         switch(month)
122         {
123                 case 1: Month = 0; break;
124                 case 2: Month = 31; break;
125                 case 3: Month = 59; break;
126                 case 4: Month = 90; break;
127                 case 5: Month = 120; break;
128                 case 6: Month = 151; break;
129                 case 7: Month = 181; break;
130                 case 8: Month = 212; break;
131                 case 9: Month = 243; break;
132                 case 10: Month = 273; break;
133                 case 11: Month = 304; break;
134                 case 12: Month = 334; break;
135         }
136         return true;
137 }
138
139
140 bool Navi3D_SunPos::setTime(string time)
141 {
142         float hours = (float)atoi(time.substr(0,time.find(":")).c_str());
143         time = time.erase(0,time.find(":")+1);
144
145         float minutes = (float)atoi(time.c_str());
146
147         Time = hours + (minutes / 60);
148
149         return true;
150 }
151
152 bool Navi3D_SunPos::setTime(int hour, int minute)
153 {
154         Time = (float)hour + (((float)minute) / 60);
155         return true;
156 }
157
158 bool Navi3D_SunPos::setTime(float time)
159 {
160         Time = time;
161         return true;
162 }
163
164 bool Navi3D_SunPos::setSunRadius(float radius)
165 {
166         Sun_Radius = radius;
167         return true;
168 }
169
170 bool Navi3D_SunPos::setHeading(float angle)
171 {
172         if(angle < 0)
173                 angle = 360 - abs(angle);
174
175         Heading = angle;
176         return true;
177 }
178
179 bool Navi3D_SunPos::calculateSunPos(void)
180 {
```

```
181         if(OldTime == Time)
182         {
183                 return true; //No Change!
184         }
185
186         OldTime = Time;
187
188         if(Timezone > 0)
189                 Sun_AngleToNorth = (15 * Time - ((c/10)*5));
190         else
191                 Sun_AngleToNorth = (15 * Time + ((c/10)*5));
192
193         int n = Month + Day;
194         float b = Latitude;
195
196         Sun_HightAngle = (float)(sin((Time-6)/12*M_PI-((float)c)/180*M_PI)*(90-b)+23.5*sin((((float)n)
                +284)/365*2*M_PI));
197
198         /*Conversion angle to radian*/
199         Rad_Sun_AngleToNorth = (float)(Sun_AngleToNorth / 180.0f * M_PI);
200         Rad_Sun_HightAngle = (float)(Sun_HightAngle / 180.0f * M_PI);
201
202         Rad_Sun_AngleToNorth = (float)((2.0f*M_PI) - Rad_Sun_AngleToNorth);
203         Rad_Sun_AngleToNorth = (float)(Rad_Sun_AngleToNorth + (0.5f * M_PI));
204         Rad_Sun_HightAngle = (float)((0.5f * M_PI) - Rad_Sun_HightAngle);
205
206         /*Conversion from polar- in cartesian coordinates*/
207         sunPos_x = Sun_Radius * sin(Rad_Sun_HightAngle) * cos (Rad_Sun_AngleToNorth);
208         sunPos_y = Sun_Radius * sin(Rad_Sun_HightAngle) * sin (Rad_Sun_AngleToNorth) * (-1);
209         sunPos_z = Sun_Radius * cos(Rad_Sun_HightAngle);
210
211         /*Adding the Heading*/
212         Rad_Heading = Heading * (float)2.0 * (float)M_PI / (float)360.0;
213
214         //Rotation (y)
215         float buffsunPos_x = sunPos_z * sin(Rad_Heading) + sunPos_x * cos(Rad_Heading);
216         float buffsunPos_y = sunPos_y;
217         float buffsunPos_z = sunPos_z * cos(Rad_Heading) - sunPos_x * sin(Rad_Heading);
218
219         sunPos_x = buffsunPos_x;
220         sunPos_y = buffsunPos_y;
221         sunPos_z = buffsunPos_z;
222
223         return true;
224 }
```

A.6.3 class Navi3D_Display.cpp

```
1  ////////////////////////////////////////////////////////////////////////////////
2  //
3  //                      Hochschule Darmstadt - SS09 - Navi3D
4  //                              Copyright by h-da
5  //
6  //                      Created: Marco Münch / 2009/04/21
7  //
8  //                      Person in charge : Marco Münch, Michael Roth
9  //
10 //                      last Edit by: Andreas Brust, 24.11.2009
11 //
12 //                      OpenGL Navi3D-Application
13 //
14 ////////////////////////////////////////////////////////////////////////////////
15
16 #pragma once
17 #include "Navi3D_InterpolatePoints.h"
18 #include "Navi3D_Base.h"
19 #include "Navi3D_SunPos.h"
```

```
#include "Navi3D_POI.h"
#include "Navi3D_Arrow.h"
#include "Define.h"

#include <windows.h>
#include <stdio.h>
#include <GL/glut.h>
#include <iostream>
#include <time.h>
#ifndef M_PI
#define M_PI    3.14159265358979323846f
#endif

#ifdef SHADER
#include "Shader.h"
#endif

using namespace std;

class Navi3D_Display{
public:
        Navi3D_Display(void);
        Navi3D_Display(int argc, char** argv, IGADA* _pGADA, Navi3D_Base* _pBase);
        ~Navi3D_Display(void);
        void Geometry();
        void switchBufferIfNew();

        float eyeX, eyeY, eyeZ;
        float Augenabstand;

        void changeArrowColor(int newColor);
        void ArrowSegmentation();
        void setNewArrowShadow(float newMaxShadow);
        void setNewBaseFactor(float factor);
        void setNewArrowProfile(int profileNr);
        void setArrowPeakType(int ArrowPeakType);
        void setArrowBegin(int ArrowBegin);
        void calculateArrow();

private:
        IGADA* pGADA;
        Navi3D_Base *pBase;
        Navi3D_SunPos *pSunpos;
        Navi3D_POI *pPOI;
        Navi3D_Arrow *pArrow;
        void init(void);
};
```

A.6.4 class Navi3D_Display.cpp

```
#include "Navi3D_Display.h"
#include "KBInterpolator.h"
#include "Textures.h"
#include "ShadowObjekts.h"

extern void printLog(string Log, unsigned int Level);

#define red 0
#define green 1
#define blue 2
#define pink 3
#define yellow 4
#define turquoise 5
#define white 6

int font=(int)GLUT_BITMAP_TIMES_ROMAN_24;
int bitmapHeight=25;
```

```
18 int winwidth;
19 int winheight;
20
21 Navi3D_Display *pDisplay;
22 float demoHead = 0.0;
23 float timeON = false;
24 float demotime = 12.0; //Demotime for SunPos
25
26 #define MIN_SUN_ANGLE 20
27
28 #define WINDOW_WIDTH 500
29 #define WINDOW_HEIGHT 500
30
31 //* Woohaaa - This is DEBUG ONLY shit...
32 Navi3D_InterpolatePoints* glutGADA;
33 KBInterpolator* glutInterpolator;
34
35 //Glut Lightsources
36 GLfloat LightAmbient[]= { 1.0f, 1.0f, 1.0f, 1.0f };  //Ambient of Lightsource
37 GLfloat LightDiffuse[]= { 1.0f, 1.0f, 1.0f, 1.0f };      //Diffuse of Lightsource
38 GLfloat LightPosition[]= { 0.0f, 100.0f, 0.0f, 1.0f };  //Position of Lightsource
39 GLfloat color[4]= {0.0f,0.0f,0.0f,0.0f};
40
41
42 #ifdef TIMER
43         clock_t oldTime = 0, newTime = 0;
44         int functionTimer =0;
45 #endif
46
47
48 void setOrthographicProjection() {
49
50         winwidth = glutGet(GLUT_WINDOW_WIDTH);
51         winheight = glutGet(GLUT_WINDOW_HEIGHT);
52
53         // switch to projection mode
54         glMatrixMode(GL_PROJECTION);
55         // save previous matrix which contains the
56         //settings for the perspective projection
57         glPushMatrix();
58         // reset matrix
59         glLoadIdentity();
60         // set a 2D orthographic projection
61         gluOrtho2D(0, winwidth, 0, winheight);
62         // invert the y axis, down is positive
63         glScalef(1, -1, 1);
64         // mover the origin from the bottom left corner
65         // to the upper left corner
66         glTranslatef(0, -winheight, 0);
67         glMatrixMode(GL_MODELVIEW);
68 }
69
70 void resetPerspectiveProjection() {
71         glMatrixMode(GL_PROJECTION);
72         glPopMatrix();
73         glMatrixMode(GL_MODELVIEW);
74 }
75
76 void renderBitmapString(float x, float y, void *font,char *string)
77 {
78
79   char *c;
80   glRasterPos2f(x, y);
81   for (c=string; *c != '\0'; c++) {
82     glutBitmapCharacter(font, *c);
83   }
84 }
```

```
85
86  void renderSpacedBitmapString(float x, float y,int spacing, void *font,char *string) {
87    char *c;
88    int x1=x;
89    for (c=string; *c != '\0'; c++) {
90          glRasterPos2f(x1,y);
91      glutBitmapCharacter(font, *c);
92          x1 = x1 + glutBitmapWidth(font,*c) + spacing;
93    }
94  }
95
96
97  void renderVerticalBitmapString(float x, float y, int bitmapHeight, void *font,char *string)
98  {
99
100   char *c;
101   int i;
102   for (c=string,i=0; *c != '\0'; i++,c++) {
103         glRasterPos2f(x, y+bitmapHeight*i);
104     glutBitmapCharacter(font, *c);
105   }
106 }
107
108
109
110 void standard(int id)
111 { switch (id) {
112         case 100: exit(0); break;
113         ///change Arrow Color
114         case 201: pDisplay->changeArrowColor(red); break;
115         case 202: pDisplay->changeArrowColor(green); break;
116         case 203: pDisplay->changeArrowColor(blue); break;
117         case 204: pDisplay->changeArrowColor(yellow); break;
118         case 205: pDisplay->changeArrowColor(turquoise); break;
119         case 206: pDisplay->changeArrowColor(pink); break;
120         case 207: pDisplay->changeArrowColor(white); break;
121         //change Arrow properties
122         case 301: pDisplay->ArrowSegmentation(); break;
123         case 310: pDisplay->setNewArrowShadow(0.0); break;
124         case 311: pDisplay->setNewArrowShadow(0.25); break;
125         case 312: pDisplay->setNewArrowShadow(0.5); break;
126         case 313: pDisplay->setNewArrowShadow(0.75); break;
127         case 314: pDisplay->setNewArrowShadow(1.0); break;
128         //change Base properties
129         case 320: pDisplay->setNewBaseFactor(1.0); break;
130         case 321: pDisplay->setNewBaseFactor(1.5); break;
131         case 322: pDisplay->setNewBaseFactor(2.0); break;
132         case 323: pDisplay->setNewBaseFactor(2.5); break;
133         //set new Arrow Profile
134         case 330: pDisplay->setNewArrowProfile(1); break;
135         case 331: pDisplay->setNewArrowProfile(2); break;
136         case 332: pDisplay->setNewArrowProfile(3); break;
137         case 333: pDisplay->setNewArrowProfile(4); break;
138         case 334: pDisplay->setNewArrowProfile(5); break;
139         //set new Arrow Peak Type
140         case 340: pDisplay->setArrowPeakType(1); break;
141         case 341: pDisplay->setArrowPeakType(2); break;
142         case 342: pDisplay->setArrowPeakType(3); break;
143         case 343: pDisplay->setArrowPeakType(4); break;
144         case 344: pDisplay->setArrowPeakType(5); break;
145         case 345: pDisplay->setArrowPeakType(6); break;
146         case 346: pDisplay->setArrowPeakType(7); break;
147         //set new Arrow begin
148         case 350: pDisplay->setArrowBegin(1); break;
149         case 351: pDisplay->setArrowBegin(2); break;
150         case 352: pDisplay->setArrowBegin(3); break;
151
```

```
152         }
153 }
154
155 void mouseMenu(void)
156 {
157         int menu, subMenuArrowColor, subMenuDebug, subMenuArrow,
158                 subMaxShadow, subBaseFactor, subArrowProfile, subArrowPeakType,
159                 subArrowBegin;
160
161         subMenuArrowColor = glutCreateMenu(standard);
162         glutAddMenuEntry("red",201);
163         glutAddMenuEntry("green",202);
164         glutAddMenuEntry("blue",203);
165         glutAddMenuEntry("yellow",204);
166         glutAddMenuEntry("turquoise",205);
167         glutAddMenuEntry("pink",206);
168         glutAddMenuEntry("white",207);
169
170         subMaxShadow = glutCreateMenu(standard);
171         glutAddMenuEntry("0.00 (no Shadow)",310);
172         glutAddMenuEntry("0.25 ",311);
173         glutAddMenuEntry("0.50 (medium Shadow)",312);
174         glutAddMenuEntry("0.75",313);
175         glutAddMenuEntry("1.00 (max. Shadow)",314);
176
177         subMenuArrow = glutCreateMenu(standard);
178         glutAddMenuEntry("Arrowsegmentation",301);
179
180         subBaseFactor = glutCreateMenu(standard);
181         glutAddMenuEntry("Factor = 1.0",320);
182         glutAddMenuEntry("Factor = 1.5",321);
183         glutAddMenuEntry("Factor = 2.0",322);
184         glutAddMenuEntry("Factor = 2.5",323);
185
186         subArrowProfile = glutCreateMenu(standard);
187         glutAddMenuEntry("Profile 1",330);
188         glutAddMenuEntry("Profile 2",331);
189         glutAddMenuEntry("Profile 3",332);
190         glutAddMenuEntry("Profile 4",333);
191         glutAddMenuEntry("Profile 5",334);
192
193         subArrowPeakType= glutCreateMenu(standard);
194         glutAddMenuEntry("Normale Pfeilspitze 1",340);
195         glutAddMenuEntry("Normale Pfeilspitze 2",341);
196         glutAddMenuEntry("Normale Pfeilspitze 3",342);
197         glutAddMenuEntry("Normale Pfeilspitze 4",343);
198         glutAddMenuEntry("2D-Spitze",344);
199         glutAddMenuEntry("Bessere Pfeilspitze 1",345);
200         glutAddMenuEntry("Bessere Pfeilspitze 2",346);
201
202         subArrowBegin= glutCreateMenu(standard);
203         glutAddMenuEntry("Start Normal 1",350);
204         glutAddMenuEntry("Start vom Boden 2",351);
205         glutAddMenuEntry("Abgerundet 3",352);
206
207
208         subMenuDebug = glutCreateMenu(standard);
209         glutAddSubMenu("Arrow",subMenuArrow);
210
211         menu = glutCreateMenu(standard);
212         glutAddSubMenu("DEBUG",subMenuDebug);
213         glutAddSubMenu("Arrowcolor",subMenuArrowColor);
214         glutAddSubMenu("Shadow",subMaxShadow);
215         glutAddSubMenu("Base Factor",subBaseFactor);
216         glutAddSubMenu("Profile",subArrowProfile);
217         glutAddSubMenu("Pfeilstart",subArrowBegin);
218         glutAddSubMenu("Pfeilspitze",subArrowPeakType);
```

```
219         glutAddMenuEntry("close",100);
220         glutAttachMenu(GLUT_RIGHT_BUTTON);
221 }
222
223 #ifdef SHADER
224         /// Shader
225         GLhandleARB shader_Pfeil; //shader program
226         GLhandleARB VS_Pfeil; //vertex shader
227         GLhandleARB PS_Pfeil; //pixel shader
228
229         GLhandleARB shader_Steinboden; //shader program
230         GLhandleARB VS_Steinboden; //vertex shader
231         GLhandleARB PS_Steinboden; //pixel shader
232
233         GLhandleARB shader_Sonne; //shader program
234         GLhandleARB VS_Sonne; //vertex shader
235         GLhandleARB PS_Sonne; //pixel shader
236
237         // Methoden für Shadowmap
238         void shadowBegin(); //Beginn eines Shadowmap Model
239         void shadowEnd();   //Ende eines Shadowmap Model
240         float proj[16];
241         float mv[16];
242         GLuint depthTex; //Datei um die Shadowmap als Textur zu speichern
243
244         //Extensions für Multitexturing
245         PFNGLACTIVETEXTUREPROC  glActiveTexture; //Active Texture setzt aktive Textureinheit
246         PFNGLMULTITEXCOORD2FPROC glMultiTexCoord2f; //setzt 2d-texturkoordinaten einer textureinheit
247         PFNGLMULTITEXCOORD3FPROC glMultiTexCoord3f; //setzt 3d-texturkoordinaten einer textureinheit
248
249         //Extension functions Shader
250         extern PFNGLCREATEPROGRAMOBJECTARBPROC          glCreateProgramObjectARB;
251         extern PFNGLCREATESHADEROBJECTARBPROC       glCreateShaderObjectARB;
252         extern PFNGLSHADERSOURCEARBPROC                 glShaderSourceARB;
253         extern PFNGLCOMPILESHADERARBPROC                        glCompileShaderARB;
254         extern PFNGLATTACHOBJECTARBPROC                 glAttachObjectARB;
255         extern PFNGLLINKPROGRAMARBPROC                          glLinkProgramARB;
256         extern PFNGLUSEPROGRAMOBJECTARBPROC             glUseProgramObjectARB;
257         extern PFNGLDELETEOBJECTARBPROC                 glDeleteObjectARB;
258         extern PFNGLUNIFORM1IARBPROC                            glUniform1iARB;
259         extern PFNGLUNIFORM1FARBPROC                            glUniform1fARB;
260         extern PFNGLUNIFORM4FARBPROC                            glUniform4fARB;
261         extern PFNGLUNIFORM4FVARBPROC                           glUniform4fvARB;
262         extern PFNGLGETUNIFORMLOCATIONARBPROC           glGetUniformLocationARB;
263         extern PFNGLVERTEXATTRIB4FARBPROC                       glVertexAttrib4fARB     ;
264         extern PFNGLGETATTRIBLOCATIONARBPROC            glGetAttribLocationARB;
265         extern PFNGLGETOBJECTPARAMETERIVARBPROC glGetObjectParameterivARB;
266         extern PFNGLGETINFOLOGARBPROC                           glGetInfoLogARB;
267         extern PFNGLVALIDATEPROGRAMARBPROC                      glValidateProgramARB;
268
269         GLuint texture[9];
270
271         void shadowBegin()
272         {
273                 glActiveTexture(GL_TEXTURE1);
274                 glMatrixMode(GL_TEXTURE);
275                 glLoadIdentity();
276                 glMatrixMode(GL_MODELVIEW);
277                 glActiveTexture(GL_TEXTURE2);
278                 glBindTexture(GL_TEXTURE_2D, depthTex);
279                 glMatrixMode(GL_TEXTURE);
280                 glLoadIdentity();
281                 glTranslatef(0.5f,0.5f,0.5f);
282                 glScalef(0.5f,0.5f,0.5f);
283                 glMultMatrixf(proj);
284                 glMultMatrixf(mv);
285                 glMatrixMode(GL_MODELVIEW);
```

```
286         }
287
288         void shadowEnd()
289         {
290                 glActiveTexture(GL_TEXTURE2);
291                 glMatrixMode(GL_TEXTURE);
292                 glLoadIdentity();
293                 glMatrixMode(GL_MODELVIEW);
294                 glActiveTexture(GL_TEXTURE0);
295         }
296 #endif
297
298 void Navi3D_Display::switchBufferIfNew(){
299         pGADA->switchBufferIfNew();
300 }
301
302 void keyboard ( unsigned char key, int x, int y )  // Create Keyboard Function
303 {
304         float newValue = 0;
305
306   switch ( key ) {
307
308     case 27:        // When Escape Is Pressed...
309       exit ( 0 );   // Exit The Program
310       break;        // Ready For Next Case
311
312         //////////////////////////Speichern
313
314         case 98: //b-Taste
315         case 65:
316            //EinstellungSpeichern();
317         break;
318
319         ////////////////////////Pfeil
320
321         case 97: //a-Taste
322         case 64:
323            //Pfeilbreite = Pfeilbreite-0.01;
324         break;
325
326         case 81://q-Taste
327         case 113:
328            //Pfeilbreite = Pfeilbreite+0.01;
329         break;
330
331         case 115: //s-Taste
332         case 83:
333            //Pfeilhoehe = Pfeilhoehe-0.01;
334         break;
335
336         case 87://w-Taste
337         case 119:
338            //Pfeilhoehe = Pfeilhoehe+0.01;
339         break;
340
341         ////////////////////////Bildschirm
342
343         case 77: //m-Taste
344         case 109:
345            glutFullScreen ( );
346         break;
347
348         case 78://n-Taste
349         case 110:
350            glutReshapeWindow ( 500, 500 );
351         break;
352
```

```
353         ////////////////////////Augenabstand
354
355         case 90: //z-Taste
356         case 122:
357            pDisplay->Augenabstand = pDisplay->Augenabstand + 0.005;
358         break;
359
360         case 85://u-Taste
361         case 117:
362            pDisplay->Augenabstand = pDisplay->Augenabstand - 0.005;
363         break;
364
365         ////////////////////////Farben
366
367         case 69: //e-Taste
368         case 101:
369
370         break;
371
372         case 68://d-Taste
373         case 100:
374
375         break;
376
377         case 82: //r-Taste
378         case 114:
379
380         break;
381
382         case 70://f-Taste
383         case 102:
384
385         break;
386
387         case 84: //t-Taste
388         case 116:
389
390         break;
391
392         case 71://g-Taste
393         case 103:
394
395         break;
396
397         /////////// Vordef. Farben
398
399         case 89://y-Taste
400         case 121:
401
402         break;
403
404         case 88: //x-Taste
405         case 120:
406
407         break;
408
409         case 67://c-Taste
410         case 99:
411
412         break;
413
414 //* Interpolator fine tuning
415 #ifdef INTERPOLATE_POINTS
416         //* i - Tension++
417         case 105:
418                 newValue = glutInterpolator->getTension() + 0.1f;
419                 glutInterpolator->setTension(newValue);
```

```
                cout << "[*] Tension = " << newValue << endl;
                break;

        //* j - Tension--
        case 106:
                newValue = glutInterpolator->getTension() - 0.1f;
                glutInterpolator->setTension(newValue);
                cout << "[*] Tension = " << newValue << endl;
                break;

        //* o - Bias++
        case 111:
                newValue = glutInterpolator->getBias() + 0.1f;
                glutInterpolator->setBias(newValue);
                cout << "[*] Bias = " << newValue << endl;
                break;

        //* k - Bias--
        case 107:
                newValue = glutInterpolator->getBias() - 0.1f;
                glutInterpolator->setBias(newValue);
                cout << "[*] Bias = " << newValue << endl;
                break;

        //* p - Continuity++
        case 112:
                newValue = glutInterpolator->getContinuity() + 100.1f;
                glutInterpolator->setContinuity(newValue);
                cout << "[*] Continuity = " << newValue << endl;
                break;

        //* l - Continuity--
        case 108:
                newValue = glutInterpolator->getContinuity() - 0.1f;
                glutInterpolator->setContinuity(newValue);
                cout << "[*] Continuity = " << newValue << endl;
                break;
#endif

    default:        // Now Wrap It Up
      break;
  }
}
void arrow_keys ( int a_keys, int x, int y )  // Create Special Function (required for arrow keys)
{
  switch ( a_keys ) {
    case GLUT_KEY_UP:     // When Up Arrow Is Pressed...
                pDisplay->eyeY = pDisplay->eyeY+0.1;
      break;
    case GLUT_KEY_DOWN:               // When Down Arrow Is Pressed...
                pDisplay->eyeY = pDisplay->eyeY-0.1;
      break;
        case GLUT_KEY_PAGE_UP:
                pDisplay->eyeZ = pDisplay->eyeZ  -0.1;
                break;
        case GLUT_KEY_PAGE_DOWN:
                pDisplay->eyeZ = pDisplay->eyeZ  +0.1;
                break;
        case GLUT_KEY_LEFT:
                pDisplay->eyeX = pDisplay->eyeX-0.1;
      break;
    case GLUT_KEY_RIGHT:
                pDisplay->eyeX = pDisplay->eyeX+0.1;
      break;
        case GLUT_KEY_F1:
                demoHead+=90;
      break;
```

```
        case GLUT_KEY_F2:
                {
                        if(timeON)
                                timeON=false;
                        else
                                timeON=true;
                }
      break;
        case GLUT_KEY_END:
                break;
    default:
      break;
  }
}


void display( void )
{
        pDisplay->switchBufferIfNew();
        pDisplay->calculateArrow();

        winwidth = glutGet(GLUT_WINDOW_WIDTH);
        winheight = glutGet(GLUT_WINDOW_HEIGHT);

#ifdef STEREOPROJECTION

        glLoadIdentity ( );


        glMatrixMode (GL_PROJECTION);
        glLoadIdentity();

        gluPerspective(45.0f,(GLfloat)(winwidth/2.0f)/
                        (GLfloat)winheight,0.1f, 3000.0f);

        // Szene fuer das linke Auge rendern (linker Viewport)
        // linke haelfte des Fensters
        glViewport (0, 0, winwidth/2,winheight);

        // Matrix sichern (fuer das rechte Auge)
        glPushMatrix();

        //glLoadIdentity ( );
                gluLookAt(pDisplay->eyeX+pDisplay->Augenabstand, pDisplay->eyeY, pDisplay->eyeZ,
                      0+pDisplay->Augenabstand, 0, -10,
                          0.0f,1.0f,0.0f);

        glMatrixMode (GL_MODELVIEW);
        glLoadIdentity();

        // Framebuffer,Depthbuffer loeschen
        glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        pDisplay->Geometry();

                // Szene fuer das rechte Auge rendern (rechter Viewport)
        glViewport (winwidth/2, 0, winwidth/2, winheight);
        glMatrixMode (GL_PROJECTION);

        // WICHTIG:
        // Kein glClear aufrufen da sonst die linke Szene
        // verlohren geht
        // Matrix vom Stack holen
        glPopMatrix();

        // Kamera fuer das rechte Auge verschieben
        //gluLookAt(auge_d/2.0,0.0,1.0, auge_d/2.0,0.0,0.0,0.0,1.0,0.0);
```

```
554         gluLookAt(pDisplay->eyeX-pDisplay->Augenabstand, pDisplay->eyeY, pDisplay->eyeZ,
555                   0-pDisplay->Augenabstand, 0, -10,
556                       0.0f,1.0f,0.0f);
557
558
559         glMatrixMode (GL_MODELVIEW);
560         glLoadIdentity();
561
562         // Aufrufen der Szenen Geometrie
563         pDisplay->Geometry();
564
565         glutSwapBuffers();
566 #else
567         glPushMatrix();
568         glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
569
570         glLoadIdentity ( );
571
572
573         glMatrixMode (GL_PROJECTION);
574         glLoadIdentity();
575
576         gluPerspective(45.0f,(GLfloat)(winwidth)/
577                        (GLfloat)winheight,0.1f, 3000.0f);
578
579         // Szene fuer das linke Auge rendern (linker Viewport)
580         // linke haelfte des Fensters
581         glViewport (0, 0, winwidth,winheight);
582
583         // Matrix sichern (fuer das rechte Auge)
584         glPushMatrix();
585
586         //glLoadIdentity ( );
587                 gluLookAt(pDisplay->eyeX+pDisplay->Augenabstand, pDisplay->eyeY, pDisplay->eyeZ,
588                       0+pDisplay->Augenabstand, 0, -10,
589                           0.0f,1.0f,0.0f);
590
591         glMatrixMode (GL_MODELVIEW);
592         glLoadIdentity();
593
594         // Framebuffer,Depthbuffer loeschen
595         glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
596
597         pDisplay->Geometry();
598
599         glutSwapBuffers();
600
601 #endif
602
603
604         #ifdef TIMER
605                 functionTimer++;
606                 if(!(functionTimer%TIMERCOUNTER))
607                 {
608                         newTime = clock()-oldTime;
609                         double time=((float)(newTime)/CLOCKS_PER_SEC);
610
611                         cout << "Timer: " << functionTimer << endl;
612                         cout << time/TIMERCOUNTER << " s pro Renderschritt" << endl;
613                         cout << TIMERCOUNTER/time << " Frames pro Sekunde" << endl;
614                         oldTime=clock();
615                 }
616         #endif
617
618 }
619
620
```

```
621
622 void Animate (int value)
623 {
624         // An dieser Stelle werden Berechnungen durchgef?hrt, die zu einer
625         // Animation der Szene erforderlich sind. Dieser Prozess l?uft im
626         // Hintergrund und wird hier alle 50 msec aufgerufen.
627
628         // RenderScene aufrufen
629         glutPostRedisplay();
630         // Timer wieder registrieren - Animate wird so wieder aufgerufen
631         glutTimerFunc(50,Animate,value);
632 }
633
634 void reshape( int w, int h )
635 {
636         // Prevent a divide by zero, when window is too short
637         // (you cant make a window of zero width).
638         if(h == 0)
639                 h = 1;
640
641         float ratio = 1.0f * w / h;
642         // Reset the coordinate system before modifying
643         glMatrixMode(GL_PROJECTION);
644         glLoadIdentity();
645
646         // Set the viewport to be the entire window
647     glViewport(0, 0, w, h);
648
649         // Set the clipping volume
650         gluPerspective(80,ratio,1,3000);
651         glMatrixMode(GL_MODELVIEW);
652         glLoadIdentity();
653 }
654
655
656 Navi3D_Display::Navi3D_Display(void)
657 {
658 }
659
660 Navi3D_Display::~Navi3D_Display(void)
661 {
662 }
663
664 void Navi3D_Display::changeArrowColor(int newColor)
665 {
666         pArrow->changeArrowColor(newColor);
667 }
668
669 void Navi3D_Display::setNewBaseFactor(float factor)
670 {
671         if(!(pArrow->setBaseFactor(factor)))
672                 printLog("Navi3D-Display.cpp : Error in setNewBaseFactor", 3);
673 }
674
675 void Navi3D_Display::ArrowSegmentation()
676 {
677         pArrow->changeArrowSegmentation();;
678 }
679
680 void Navi3D_Display::setNewArrowShadow(float newMaxShadow)
681 {
682         pArrow->setMaxShadow(newMaxShadow);
683 }
684
685 void Navi3D_Display::setNewArrowProfile(int profileNr)
686 {
687         if(!(pArrow->setNewArrowProfile(profileNr)))
```

```
688                 printLog("Navi3D-Display.cpp : Error in setNewArrowProfile", 3);
689 }
690
691 void Navi3D_Display::setArrowBegin(int ArrowBegin)
692 {
693         if(!(pArrow->setArrowBegin(ArrowBegin)))
694                 printLog("Navi3D-Display.cpp : Error in setArrowBegin", 3);
695 }
696
697 void Navi3D_Display::setArrowPeakType(int ArrowPeakType)
698 {
699         if(!(pArrow->setArrowPeakType(ArrowPeakType)))
700                 printLog("Navi3D-Display.cpp : Error in setArrowPeakType", 3);
701
702 }
703
704 void Navi3D_Display::calculateArrow()
705 {
706         if(!(pArrow->calculateArrowData()))
707                 printLog("Navi3D-Display.cpp : Error in calculateArrow", 3);
708 }
709
710 Navi3D_Display::Navi3D_Display(int argc, char** argv, IGADA* _pGADA, Navi3D_Base* _pBase)
711 {
712         //
713         Augenabstand=0.01f;
714         eyeX = 0;
715         eyeY = 4;
716         eyeZ = 9;
717
718         //* Set DEBUG pointers to manipulate interpolator during runtime
719         #ifdef INTERPOLATE_POINTS
720                 glutGADA = static_cast<Navi3D_InterpolatePoints*>(_pGADA);
721                 glutInterpolator = static_cast<KBInterpolator*>(glutGADA->getInterpolator());
722         #endif
723                 //newValue = static_cast<Navi3D_InterpolatePoints*>(glutGADA)->getTension() + 0.1f;
724
725         //Create and set Pointers
726         pDisplay = this;
727         pGADA = _pGADA;
728         pBase = _pBase;
729         pSunpos = new Navi3D_SunPos();
730         pPOI = new Navi3D_POI(_pBase, _pGADA);
731         pArrow = new Navi3D_Arrow(_pBase, _pGADA);
732
733         //Glut Functions
734         glutInit(&argc, argv); // Erm Just Write It =)
735         glutInitDisplayMode ( GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA ); // Display Mode
736         glutInitWindowPosition (0,0);
737         glutInitWindowSize  ( WINDOW_WIDTH, WINDOW_HEIGHT ); // If glutFullScreen wasn't called this
                is the window size
738         glutCreateWindow    ( "Navi3D" ); // Window Title (argv[0] for current directory as title)
739         init ();
740         glutFullScreen      (  );          // Put Into Full Screen
741
742         #ifdef TIMER
743                 oldTime = clock();
744         #endif
745
746         glutDisplayFunc     ( display );  // Matching Earlier Functions To Their Counterparts
747         glutReshapeFunc     ( reshape );
748         glutKeyboardFunc    ( keyboard );
749         glutSpecialFunc     ( arrow_keys );
750         glutIdleFunc            ( display );
751
752         glutTimerFunc(50,Animate,0);
753
```

```
754         mouseMenu();
755
756         glutMainLoop        ( );          // Initialize The Main Loop*/
757 }
758
759 void Navi3D_Display::init(void)
760 {
761         //activate Depth Test
762         glEnable(GL_DEPTH_TEST);
763         glDepthFunc(GL_LEQUAL);
764         glClearDepth(1.0f);
765
766     glClearColor(0.0f, 0.0f, 0.0f, 0.5f);                                   // Black Background
767
768         #ifdef BECKER_MODE
769                         pSunpos->setDate("05.05.2009");
770                         pSunpos->setLatitude((float)49.94);
771                         pSunpos->setLongitude((float)8.74);
772                         pSunpos->setSunRadius(20);
773                         pSunpos->setTime(demotime);
774                         pSunpos->setTimezone(0.0);
775                         pSunpos->setHeading(180.0);
776         #else
777                         pSunpos->setDate(pGADA->getMonth(),pGADA->getDay());
778                         pSunpos->setLatitude((float)pGADA->getLatitude()/1000.0);
779                         pSunpos->setLongitude((float)pGADA->getLongitude()/1000.0);
780                         pSunpos->setTime(pGADA->getUtc_hour(),pGADA->getUtc_minute());
781                         pSunpos->setTimezone(1.0);
782                         pSunpos->setSunRadius(10);
783                         pSunpos->setHeading(pGADA->getHeading());//ohne Target geht das wohl nicht!
784         #endif
785
786         #ifdef SHADER
787                 glGenTextures(1, &depthTex);
788                 glBindTexture(GL_TEXTURE_2D, depthTex);
789                 glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT,512,512, 0,GL_DEPTH_COMPONENT,
                        GL_UNSIGNED_BYTE, 0);
790                 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
791                 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
792                 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
793                 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
794                 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_COMPARE_MODE, GL_COMPARE_R_TO_TEXTURE);
795
796                 //Adressen der Multitexturing-Funktionen setzen
797                 glActiveTexture = (PFNGLACTIVETEXTUREPROC) wglGetProcAddress("glActiveTexture");
798                 glMultiTexCoord2f = (PFNGLMULTITEXCOORD2FPROC) wglGetProcAddress("glMultiTexCoord2f");
799                 glMultiTexCoord3f = (PFNGLMULTITEXCOORD3FPROC) wglGetProcAddress("glMultiTexCoord3f");
800
801                 //Texturen laden
802                 CreateTGATexture(&texture[5], "Textures//Steinboden.tga");
803                 CreateTGATexture(&texture[6], "Textures//Sonne.tga");
804
805                 //glClearColor(1.0f, 1.0f, 1.0f, 0.0f);
806                 glEnable(GL_DEPTH_TEST);
807                 glDepthFunc(GL_LEQUAL);
808                 glClearDepth(1.0f);
809
810                 //Shader initialisieren
811                 bool glslSupported = InitGLSL();
812                 if(!glslSupported)
813                         cout<<"Error Initializing GLSL"<<endl;
814                 else
815                         cout<<"Success Initializing GLSL"<<endl;
816
817                 //Shader laden
818                 LoadShader(&shader_Pfeil, &VS_Pfeil, &PS_Pfeil, "Shader//VS_Pfeil.txt", "Shader//
                        PS_Pfeil.txt");
```

```
819                 LoadShader(&shader_Steinboden, &VS_Steinboden, &PS_Steinboden, "Shader//VS_Steinboden.
                        txt", "Shader//PS_Steinboden.txt");
820                 LoadShader(&shader_Sonne, &VS_Sonne, &PS_Sonne, "Shader//VS_Sonne.txt", "Shader//
                        PS_Sonne.txt");
821         #else
822                 glEnable(GL_NORMALIZE);
823                 glEnable(GL_LIGHTING); // Activate Light
824                 glShadeModel(GL_SMOOTH); //Activate Shading
825                 //set Lights
826                 //glLightfv(GL_LIGHT1, GL_AMBIENT, LightAmbient);                       // Setup The
                        Ambient Light
827                 glLightfv(GL_LIGHT1, GL_DIFFUSE, LightDiffuse);                       // Setup The
                        Diffuse Light
828                 glLightfv(GL_LIGHT1, GL_POSITION,LightPosition);                      // Position
                        The Light
829                 glEnable(GL_LIGHT1);
830
831                 glEnable(GL_COLOR_MATERIAL); //Enable the Color Material
832         #endif
833
834         #ifdef FOG
835                 glFogfv(GL_FOG_COLOR, color);
836                 glFogf(GL_FOG_START, FOG_START);
837                 glFogf(GL_FOG_END, FOG_END);
838                 glFogi(GL_FOG_MODE, GL_LINEAR);
839                 glEnable(GL_FOG);
840         #endif
841 }
842
843 void Navi3D_Display::Geometry()
844 {
845         #ifdef BECKER_MODE
846                 /*Demo for Sunpos*/
847                 if(timeON)
848                         demotime += (float)0.05;
849
850                 if(demotime > 19)
851                         demotime = 5.0;
852
853                 pSunpos->setTime(demotime);
854
855                 if(demoHead > 360.0)
856                         demoHead=0.0;
857
858                 pSunpos->setHeading(demoHead);
859         #else
860                 pSunpos->setDate(pGADA->getMonth(),pGADA->getDay());
861                 pSunpos->setLatitude((float)pGADA->getLatitude()/1000.0);
862                 pSunpos->setLongitude((float)pGADA->getLongitude()/1000.0);
863                 pSunpos->setTime(pGADA->getUtc_hour(),pGADA->getUtc_minute());
864                 pSunpos->setTimezone(1.0);
865                 pSunpos->setSunRadius(10);
866                 pSunpos->setHeading(pGADA->getHeading());//ohne Target geht das wohl nicht!
867         #endif
868
869         #ifdef SHADER
870                 ////////////////////////////////////////
871                 // Shadowmap erzeugen
872                 ////////////////////////////////////////
873                 glViewport(0,0,512,512); //Setzt den Viewport auf 512x512 die größe der Shaowmap
                        Textur
874                 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //Leert die Buffer
875                 glLoadIdentity();
876                 gluLookAt(pSunpos->getSunPosX(), pSunpos->getSunPosY(), pSunpos->getSunPosZ(), 0.0015f
                        ,0.001f,0.005f, 0,1,0); //setzt die Kamera auf Poistion der Lichtquelle
877
878                 glGetFloatv(GL_MODELVIEW_MATRIX, mv);
```

```
879                 glGetFloatv(GL_PROJECTION_MATRIX, proj);
880
881                 pArrow->Arrow_Display();
882
883                 glBindTexture(GL_TEXTURE_2D, depthTex); //Bindet die Textur
884                 glCopyTexImage2D(GL_TEXTURE_2D, 0,GL_DEPTH_COMPONENT, 0,0,512,512,0); //Speichert das
                        Bild in die depthTex Textur
885
886         #endif
887
888                 glViewport (0, 0, winwidth,winheight); //setzt den Viewport wieder zurück
889                 glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //Leert die Buffer
890                 glLoadIdentity();
891                 gluLookAt(pDisplay->eyeX+pDisplay->Augenabstand, pDisplay->eyeY, pDisplay->eyeZ,
892                                 0+pDisplay->Augenabstand, 0, -10,
893                                 0.0f,1.0f,0.0f);
894         #ifdef SHADER
895                 glUseProgramObjectARB(shader_Sonne);
896
897                 if(pSunpos->getSunHightAngle() > MIN_SUN_ANGLE)
898                 {
899                         //Objekte Tag
900                         Sun(pSunpos->getSunPosX(),pSunpos->getSunPosY(),pSunpos->getSunPosZ());
901                 }
902                 else
903                 {
904                         //Objekte Nacht (z.B. Mond)
905                 }
906
907                 glUseProgramObjectARB(shader_Steinboden); //Initialisiert den Shader für den
                        Steinboden
908
909                 shadowBegin();
910                         int uniform = glGetUniformLocationARB(shader_Steinboden, "depthTex");
911                         glUniform1iARB(uniform, 2); //erstes bild kommt in erste textureinheit
912
913                         uniform = glGetUniformLocationARB(shader_Steinboden, "lightPos");
914                         glUniform4fARB(uniform, pSunpos->getSunPosX(), pSunpos->getSunPosY(), pSunpos
                                ->getSunPosZ(), 1.0f);
915
916                         glActiveTexture(GL_TEXTURE2);
917                         glBindTexture(GL_TEXTURE_2D, depthTex); //...und die in die textur gerenderte
                                szene binden
918
919                 if(pSunpos->getSunHightAngle() > MIN_SUN_ANGLE)
920                 {
921                         //Tag wirft Schatten
922                         Ground();
923                 }
924                 shadowEnd();
925
926                 glUseProgramObjectARB(shader_Pfeil);
927         #endif
928
929         pArrow->Arrow_Display();
930
931         glDisable(GL_LIGHT1);
932                 if(pSunpos->getSunHightAngle() > MIN_SUN_ANGLE)
933                 {
934                         // Licht Tag
935                         LightPosition[0] = pSunpos->getSunPosX();
936                         LightPosition[1] = pSunpos->getSunPosY();
937                         LightPosition[2] = pSunpos->getSunPosZ();
938
939                         glLightfv(GL_LIGHT1, GL_POSITION,LightPosition);                        //
                                Position The Light
940                 }
```

```
941                 else
942                 {
943                         //Licht Nacht
944                 }
945         glEnable(GL_LIGHT1);
946
947         cout << pSunpos->getSunPosX() << "  " << pSunpos->getSunPosY() << "  "<< pSunpos->getSunPosZ()
                 << "  " << endl;
948
949         #ifdef SHADER
950                 glUseProgramObjectARB(0);
951         #endif
952
953         //pPOI->DisplayPOI();
954
955         glPopMatrix();
956
957                 // Display sentence
958         glDisable(GL_LIGHTING);
959                 glPushMatrix();
960                         glColor3f(1.0f,1.0f,1.0f);
961                         setOrthographicProjection();
962                         glLoadIdentity();
963 #ifdef COLOR_ON_ARROW_SPEEDCONTROL
964                         renderSpacedBitmapString((winwidth/2)-150,winheight-100,10,(void *)font,pArrow
                             ->getNextCurveInMeter());
965 #endif
966                         resetPerspectiveProjection();
967                         glPopMatrix();
968                 glPopMatrix();
969         glEnable(GL_LIGHTING);
970 }
```

A.6.5 Vertex Shader Phong

```
1  varying vec3 normal;
2  varying vec3 v;
3  varying vec3 lightvec;
4  void main(void)
5  {
6          normal          = normalize(gl_NormalMatrix * gl_Normal);
7          v               = vec3(gl_ModelViewMatrix * gl_Vertex);
8          lightvec        = normalize(gl_LightSource[1].position.xyz - v);
9          gl_Position     = gl_ModelViewProjectionMatrix * gl_Vertex;
10
11         gl_FrontColor = gl_Color;
12 }
```

A.6.6 Pixel Shader Phong

```
1  varying vec3 normal;
2  varying vec3 v;
3  varying vec3 lightvec;
4
5  void main(void)
6  {
7          vec3 Eye             = normalize(-v);
8
9          vec3 Reflected       = normalize( reflect( -lightvec, normal ));
10
11         vec4 IAmbient        = gl_LightSource[0].ambient;
12         vec4 IDiffuse        = gl_LightSource[0].diffuse * max(dot(normal, lightvec), 0.0);
13         vec4 ISpecular       = gl_LightSource[0].specular * max(dot(Reflected, Eye), 0.0);
14
15         gl_FragColor = gl_Color * (IDiffuse + IAmbient + pow(ISpecular,16) );
16 }
```

A.6.7 Vertex Shader Steinboden

```
uniform vec4 lightPos;
uniform vec4 cameraLoc;

varying vec3 normal;
varying vec3 lightVec;
varying vec3 halfVec;
varying vec4 projCoords;

void main()
{
        gl_Position = ftransform();
        normal = normalize(gl_NormalMatrix * gl_Normal);
        vec4 temp = (gl_LightSource[1].position)- (gl_ModelViewMatrix * gl_Vertex);
        lightVec = normalize(temp.xyz);

        //Specular
        vec3 camDir = normalize((gl_ModelViewMatrix *cameraLoc)- (gl_ModelViewMatrix * gl_Vertex)).xyz
            ;
        halfVec = normalize(camDir + lightVec);

        //ftransform == gl_ModelViewProjectionMatrix * gl_Vertex;
        gl_TexCoord[0] = gl_MultiTexCoord0;

        ////////////////////////////////
        projCoords = gl_TextureMatrix[2]*(gl_TextureMatrix[1]*gl_Vertex);
        ////////////////////////////////

}
```

A.6.8 Pixel Shader Steinboden

```
varying vec3 normal;
varying vec3 lightVec;
varying vec3 halfVec;
varying vec4 projCoords;

uniform sampler2DShadow depthTex;

void main (void)
{
    vec3 shadowCoords = projCoords.xyz/projCoords.w;
    shadowCoords.z -= 0.00003;
    shadowCoords.xy = clamp(shadowCoords.xy, 0.01, 0.99);
    vec4 shadow = shadow2D(depthTex, shadowCoords);

    if(shadow == 1.0)
    {
        gl_FragColor = vec4(0.0,0.0,0.0,1.0)*(shadow);
    }
    else
    {
        gl_FragColor = vec4(0.5,0.5,0.5,1.0);
    }
}
```

A.6.9 Vertex Shader Pfeil

```
uniform vec4 lightPos;
uniform vec4 cameraLoc;

varying vec3 normal;
varying vec3 lightVec;
varying vec3 halfVec;

```

```
attribute vec3 tangent;

void main()
{

        normal = normalize(gl_NormalMatrix * gl_Normal);
        vec4 temp = (gl_LightSource[1].position) - (gl_ModelViewMatrix * gl_Vertex);
        lightVec = normalize(temp.xyz);

        //Specular
        vec3 camDir = normalize((gl_ModelViewMatrix *cameraLoc)- (gl_ModelViewMatrix * gl_Vertex)).xyz
            ;
        halfVec = normalize(camDir + lightVec);

        gl_Position = ftransform();

        gl_FrontColor = gl_Color;
}
```

A.6.10 Pixel Shader Pfeil

```
varying vec3 normal;
varying vec3 lightVec;
varying vec3 halfVec;

void main()
{
        vec3 n,halfV;
        vec4 buff;
        float NdotL, NdotHV;

        vec4 color = vec4(0.0,0.0,0.0,0.0);
        n = normalize(normal);
        NdotL = max(dot(n,lightVec),0.0);

        if (NdotL > 0.0)
        {
                color += NdotL * gl_LightSource[0].diffuse.r;
                halfV = normalize(halfVec);
                NdotHV = max(dot(n,halfV),0.0);
                color += gl_LightSource[0].specular.r*pow(NdotHV,100.0);
        }
        gl_FragColor = color*gl_Color;
}
```

B Target - Codedokumentation

B.1 Protokollierung zur Portierung des Frameworks

Zunächst wurde die Protokollierung der durchgeführten Änderungen sehr allgemein gehalten. Im Folgenden sind zunächst die Pfade und Namen der geänderten Dateien sowie eine Beschreibung der Änderungen zu finden. Im weiteren Verlauf wurden die Änderungen detaillierter Protokolliert. Bei Bedarf lassen sich sämtliche Änderungen auch sehr detailliert in Perforce betrachten und nachvollziehen.

Christian Bartel

Um die Kompilierbarkeit der Windows- und QNX-Version nicht zu beeinträchtigen und unangetastet zu lassen, wurde eine neue Direktive für Linux eingefügt (define). Diese ist in den Jamrules zu finden unter (Pfad und Dateiname):

//depot/Framework/DEV/Framework

Jamrules (ohne Endung)

In den folgenden Dateien wurden vornehmlich Include-Anweisungen angepasst. Unter anderem musste oft die Version von string, iostream und stdlib ausgetauscht werden.

Pfad und Dateiname:

//depot/Framework/DEV/Framework/src/dataContainer
CAmpDC.cpp

//depot/Framework/DEV/Framework/src/base/socket
CInetAddr.cpp

//depot/Framework/DEV/Framework/src/base
CMessage.h

//depot/Framework/DEV/Framework/src/dataContainer
CTunerDC.cpp

//depot/Framework/DEV/Framework/src/base/socket
CInetAddr.cpp

//depot/Framework/DEV/Framework/src/dataContainer
CTunerDataContainer.cpp

//depot/Framework/DEV/Framework/src/most/CDDevice
CCDDeviceHALLinux.cpp

//depot/Framework/DEV/Framework/src/most
CMostMsgReceiverThread.cpp

//depot/Framework/DEV/Framework/src/most/socketdebugger
SockServer.cpp

//depot/Framework/DEV/Framework/src/admin
CContext.cpp

//depot/Framework/DEV/Framework/src/admin
CContext.cpp

//depot/Framework/DEV/Framework/src/controller/gada
CGadaDC.cpp

//depot/Framework/DEV/Framework/src/controller/navi
CNaviDataContainer.cpp

Im Folgenden kam es zu Problemen mit den Funktionen “utoa”’, “strnset” und “getprio”. Die Fehlermeldung hat einen nicht definierten Gültigkeitsbereicht beanstandet, obwohl dieser korrekt festgelegt war. Das Problem wurde lediglich innerhalb zwei Methoden beanstandet, obwohl die Funktionen auch

an anderen Stellen der Datei benutzt werden. Trotz umfangreicher Lösungsversuche und Recherche zu diesem Problem ist es nicht gelungen den Fehler zu beheben. Nach Rücksprache mit der Systemgruppe wurde entschieden, diese beiden Methoden auszukommentieren. Sie sind für die Lauffähigkeit nicht relevant, da sie lediglich dem überprüfen der Prozessorlast bei bestimmten Anwendungen dienen. Ein FIXME-Eintrag mit Verweis auf die entsprechenden Funktionen wurde gemacht.

Pfad und Dateiname:

//depot/Framework/DEV/Framework/src/admin
CAdminComponent.cpp

Methoden:

CAdminComponent::checkProcessorLoad

CAdminComponent::forceCoreDump

```
#ifdef LINUX    //Verzweigung der Linux-Direktive

// Hier Anpassungen für Linux vornehmen

#elif WIN32
#else

void CAdminComponent::checkProcessorLoad(const char* componentName)
{

// FIXME: Funktionen ‘‘utoa’’ und ‘‘strnset’’
// dieser Methode kompilieren nicht unter Linux

[...] //fuer Dokumentation gekuerzt

}

[...] //fuer Dokumentation gekuerzt

Int8 CAdminComponent::forceCoreDump(const char* componentName)
{
// FIXME: Funktionen ‘‘utoa’’ und ‘‘getprio’’
// dieser Methode kompilieren nicht unter Linux
/*
 *  Komponente der proecess-id zuordnen
 */
UInt8 tid=0;
[...] //fuer Dokumentation gekuerzt
```

}

Die nächste Fehlermeldung bezog sich auf die Datei CMMDataLine.h und ließ sich letztendlich auf ein Abhängigkeitsproblem zurückführen. In der genannten Datei musste das include der CMMTile.h - Headerdatei auskommentiert werden:

// #include “CMMTile.h”

und CMMTile direkt als Klasse deklariert werden:

class CMMTile;

Durch diese änderungen musste die entsprechende cpp-Datei (CMMDataLine.cpp) angepasst, und dort CMMTile.h eingebunden werden.

#include “CMMTile.h”

In derselben Datei (CMMDataLine.cpp) wurde DBLine als struct deklariert:

struct DBLine *m_pLine;

In der Datei CMMDateArea.cpp musste ebenfalls noch CMMTile.h eingebunden werden:

#include “CMMTile.h”

In den folgenden Dateien wurde ein zusätzliches define für Linux eingefügt, welche die Bibliothek <new> betrifft.

#ifdef LINUX //Linux-Verzweigung

#include <new>

#elif WIN32

#else

#include <new.h>

#endif

src/gl_hmi/ss/navi/navi
CMMMapMetaData.cpp

src/gl_hmi/ss/navi/navi
CMMDataObjectFactory.cpp

src/controller/navi/routing
CNaviRoutingController.cpp

src/controller/wlanClient/socket
CSocketClient.cpp

src/gl_hmi/ss/navi/navi
database.cpp

Der nächste Fehler bezieht sich auf die Bibliothek process.h, welche Funktionen für Threads und Prozesse beinhaltet. Diese Bibliothek funktioniert nur für Windows und ist für Linux nicht erhältlich. Glücklicherweise scheinen keine Funktionen aus der process.h - Bibliothek aufgerufen oder verwendet zu werden. Das Einfügen eines leeren Linux-defines behebt diesen Fehler somit.

src/controller/debugger
CDebuggerComponent.cpp

#ifdef LINUX //Linux-Verzweigung

// FIXME: process.h ist windows-only !!

#elif WIN32
#else

#include <process.h>

#endif

In den folgenden Dateien wurde das Include der Bibliothek string.h eingefügt:

src/gl_hmi/ss/navi/navi
database.cpp

src/gl_hmi/base
CHMIImageManager.cpp

#include <string.h>

OpenGL wurde in der folgenden Datei nicht gefunden und wurde entsprechend eingebunden

src/gl_hmi/ss/navi/navi
MapControl.cpp

#include <GL/glu.h>

In derselben Datei musste vor den Funktionen ätanünd äbs"der Verweis auf die Basisklasse entfernt werden.

src/gl_hmi/ss/navi/navi
MapControl.cpp

rotyangle[rotationCacheIndex] = atan(abs(yDiff)/abs(xDiff)); "std::" vor atan und abs entfernt.

Die letzte Fehlermeldung beanstandete einen nicht definierten Gültigkeitsbereich für glOrthof, eine Funktion, die einen orthogonalen 2D-Modus aktiviert und somit die Z-Koordinate keinen Einfluss mehr auf die Größe eines Objekts nimmt [Wik10a].

src/gl_hmi/ss/navi/navi/MapControl.cpp:218: Fehler: “glOrthof” wurde in diesem Gültigkeitsbereich nicht definiert

Hier musste lediglich das “f” am Ende von glOrthof entfernt werden.

```
#ifdef LINUX //Linux-Verzweigung

glOrtho(0.0f, 800.0f, 0.0f, 480.0f, 0.0f, 1.0f);

#elif WIN32
#else

glOrthof(0.0f, 800.0f, 0.0f, 480.0f, 0.0f, 1.0f);

#endif
```

Wie bereits eingangs erwähnt, lassen sich in Perforce sämtliche Änderungen detailliert betrachten und nachvollziehen. Nach dem erfolgreichen Kompilieren wird das Framework im Ordner /bin/linux mit “./Framework” gestartet. Es erscheint folgende Ausgabe mit einigen Zeilen Debug-Print:

```
istchbart@ICM2006-6o: /depot/Framework/DEV/Framework/bin/linux$ ./Framework NG3

Version No.: D1

Admin: Creating Contexts...
Creating Context #0 for MainDispatcherComponent
Creating Context #1 for RemoteDebugComponent
Creating Context #2 for AudioMasterComponent
Creating Context #3 for MaxiComComponent
Creating Context #4 for MaxiComDeviceComponent
Creating Context #5 for NaviComponent
Creating Context #6 for AmpPonComponent
Creating Context #7 for AmpNG3Component
Creating Context #8 for TunerComponent
Creating Context #9 for CDComponent
Creating Context #10 for CDDeviceComponent
Creating Context #11 for IPodComponent
Creating Context #12 for wLanClientComponent
Creating Context #13 for wLanServerComponent
Creating Context #14 for SpeechRecComponent
Creating Context #15 for HMIComponent
Creating Context #16 for DebuggerComponent
Creating Context #17 for GadaComponent
```

Creating Context #18 for AdminComponent

CNaviDataContainer: Constructor!

CNaviGpsDataContainer: Constructor!

Admin: Contexts created.
Debugger Component started
CNaviComponent: Construktor
CHMISpeechRecErrorPanel::CHMISpeechRecErrorPanel(CHMISubsystemBase & subsystem)
CHMISpeechRecSS::CHMISpeechRecSS()
Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZäöü
Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZäöü
Alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZäöü
Alphabet: 1234567890
CDataBase::Open: Opening DB 'map/tree.dat'.
CDataBase::Open: Loading Tree.
CRaodMapNode::Load: Tile '00' loaded!
#Nodes: 669, #Edges: 645, #Faces: 886
CGadaComponent started
CRaodMapNode::Load: Tile '010' loaded!
#Nodes: 271, #Edges: 340, #Faces: 276
CRaodMapNode::Load: Tile '011' loaded!
#Nodes: 169, #Edges: 219, #Faces: 347
CRaodMapNode::Load: Tile '012' loaded!
#Nodes: 993, #Edges: 1288, #Faces: 660
CRaodMapNode::Load: Tile '013' loaded!
#Nodes: 216, #Edges: 326, #Faces: 294
CRaodMapNode::Load: Tile '020' loaded!
#Nodes: 173, #Edges: 123, #Faces: 375
CRaodMapNode::Load: Tile '021' loaded!
#Nodes: 952, #Edges: 1184, #Faces: 540
CRaodMapNode::Load: Tile '022' loaded!
#Nodes: 17, #Edges: 10, #Faces: 104
CRaodMapNode::Load: Tile '023' loaded!
#Nodes: 481, #Edges: 619, #Faces: 446
CRaodMapNode::Load: Tile '030' loaded!
#Nodes: 917, #Edges: 1125, #Faces: 626
CRaodMapNode::Load: Tile '031' loaded!
#Nodes: 99, #Edges: 136, #Faces: 183
CRaodMapNode::Load: Tile '032' loaded!
#Nodes: 177, #Edges: 196, #Faces: 381
CRaodMapNode::Load: Tile '033' loaded!
#Nodes: 0, #Edges: 0, #Faces: 0
Loaded Map! Size needed: 677468
HMI Component started

pidMaxiCom: 0

istchbart@ICM2006-6o: /depot/Framework/DEV/Framework/bin/linux$

Christian Bartel

B.2 Systemgruppe

B.2.1 CGPS_Component

In der CGPS_Component werden die vom USB-GPS-Empfänger empfangen GPS-Datensätze in einzelne Daten zerlegt und über das Messaging-System bzw. über Datencontainer des Frameworks anderen Komponenten zugänglich gemacht.
In der Methode *parseGPS* werden von den insgesamt 13 verschiedenen NMEA-0183-Datensätze [uDMW10], die als Zeichenketten vorliegen, die Datensätze *GPRMC*, *GPGGA*, *GPVTG*, geparst und die derzeit benötigten Daten extrahiert. Die extrahierten Daten werden mittels der Struktur *GpsInfoData* in den GPS-Datencontainer geschrieben.

```
1  static void parseGPS(const char* msg, GpsInfoData *gid) {
2          char buf[256];
3
4          memset(buf, 0, sizeof(buf));
5          memset(gid, 0, sizeof(gid));
6
7          /* ############## $GPRMC Message #############
8           *
9           */
10
11         char* pStart = const_cast<char *> (strstr(msg, "$GPRMC")); // bad !!
12         if (pStart != NULL) {
13                 char* pEnd = strstr(pStart, "\n");
14                 if (pEnd != NULL && pEnd - pStart < (int) sizeof(buf)) {
15                         strncpy(buf, pStart, pEnd - pStart);
16                         buf[pEnd - pStart] = '\0';
17
18                         // now tokenize for determining the contained values
19                         char token[TOKENMAXSIZE];
20                         const char* tosearch = buf; // search pointer
21                         const char* nextStart = NULL;
22
23                         size_t counter = 0;
24                         while ((nextStart = tokenize(tosearch, token, ',')) != NULL) {
25                                 switch (counter) {
26                                 case 0: //$GPRMC
27                                         // Not used
28                                         break;
29                                 case 1: // time
30                                         gid->utc_hour = (int) (atoi(token) / 10000);
31                                         gid->utc_minute = (int) (atoi(token) / 100)     - (gid->
                                                utc_hour * 100);
32                                         gid->utc_second = atoi(token) - (gid->utc_hour * 10000) - (gid
                                                ->utc_minute * 100);
33                                         break;
34                                 case 2: // Valid Flag
35                                         // Not used
36                                         break;
37                                 case 3: // latitude
38                                         //gid->latitude = convertNMEAposToDec(token, POS_LAT);
39                                         gid->latitude = atof(token);
40                                         break;
41                                 case 4: // latitude direction
```

```
                                        // Not used
                                        break;
                                case 5: // longitude
                                        //gid->longitude = convertNMEAposToDec(token, POS_LON);
                                        gid->longitude = atof(token);
                                        break;
                                case 6: // longitude direction
                                        // Not used
                                        break;
                                case 7: // velocity
                                        // Not used
                                        // IN $GPVTG it is in km/h
                                        break;
                                case 8: // direction
                                        // Not used
                                        break;
                                case 9: // date
                                        gid->day = (int) (atoi(token) / 10000);
                                        gid->month = (int) (atoi(token) / 100) - (gid->day * 100);
                                        gid->year = (atoi(token) - (gid->day * 10000) - (gid->month
                                                * 100)) + 2000;
                                        break;
                                case 10: // Misswise
                                        // Not used
                                        break;
                                case 11: // Misswise direction
                                        // Not used
                                        break;
                                default:
                                        DEBUG_TRACE("not able to handle position %d for elemnent %s",
                                             counter, token);
                                        break;
                                } // switch
                                tosearch = nextStart;
                                ++counter;
                        } // while
                } // if
        }



        /* ############ GPGGA ############
         *
         */
        pStart = const_cast<char *> (strstr(msg, "$GPGGA")); // bad !!
        if (pStart != NULL) {
                char* pEnd = strstr(pStart, "\n");
                if (pEnd != NULL && pEnd - pStart < (int) sizeof(buf)) {
                        strncpy(buf, pStart, pEnd - pStart);
                        buf[pEnd - pStart] = '\0';

                        // now tokenize for determining the contained values
                        char token[TOKENMAXSIZE];
                        const char* tosearch = buf; // search pointer
                        const char* nextStart = NULL;

                        size_t counter = 0;
                        while ((nextStart = tokenize(tosearch, token, ',')) != NULL) {
                                switch (counter) {
                                case 0: // $GPGGA
                                        // Not used
                                        break;
                                case 1: // Time
                                        // Not used
                                        break;
                                case 2: //latitude direction
                                        // Not used
```

```
107                                     break;
108                             case 3: // latitude direction
109                                     // Not used
110                                     break;
111                             case 4: // longditude
112                                     // Not used
113                                     break;
114                             case 5: // longditude direction
115                                     // Not used
116                                     break;
117                             case 6:// quality
118                                     // Not used
119                                     break;
120                             case 7: // number of satelites
121                                     // Not used
122                                     break;
123                             case 8: //horizontal dilution
124                                     // Not used
125                                     break;
126                             case 9: // height
127                                     gid->height = (int)atoi(token);
128                                     break;
129                             case 10: // anything
130                                     // Not used
131                                     break;
132                             case 11: // height geoid
133                                     // Not used
134                                     break;
135                             case 12: // anything
136                                     // Not used
137                                     break;
138                             case 13: // Null
139                                     // Not used
140                                     break;
141                             default:
142                                     DEBUG_TRACE("not able to handle position %d for elemnent %s",
                                             counter, token);
143                                     break;
144                             } // switch
145                             tosearch = nextStart;
146                             ++counter;
147                     } // while
148             } // if
149     }
150
151     /* ############ $GPVTG ############
152      *
153      */
154
155     pStart = const_cast<char *> (strstr(msg, "$GPVTG")); // bad !!
156     if (pStart != NULL) {
157             char* pEnd = strstr(pStart, "\n");
158             if (pEnd != NULL && pEnd - pStart < (int) sizeof(buf)) {
159                     strncpy(buf, pStart, pEnd - pStart);
160                     buf[pEnd - pStart] = '\0';
161
162                     // now tokenize for determining the contained values
163                     char token[TOKENMAXSIZE];
164                     const char* tosearch = buf; // search pointer
165                     const char* nextStart = NULL;
166
167                     size_t counter = 0;
168                     while ((nextStart = tokenize(tosearch, token, ',')) != NULL) {
169                             switch (counter) {
170                             case 0: //$GPVTG
171                                     //not used
172                                     break;
```

```
173                                 case 1: //course
174                                         //not used
175                                         break;
176                                 case 2: //anything
177                                         //not used
178                                         break;
179                                 case 3: //course magnetic
180                                         //not used
181                                         gid->heading = atof(token);
182                                         break;
183                                 case 4: //anything
184                                         //not used
185                                         break;
186                                 case 5: //groundspeed knots
187                                         //not used
188                                         break;
189                                 case 6: //speed direction
190                                         //not used
191                                         break;
192                                 case 7: //groundspeed km/h
193                                         gid->speed = (float)atof(token);
194                                         break;
195                                 case 8: // K for Kilometers
196                                         //not used
197                                         break;
198                                 default:
199                                         DEBUG_TRACE("not able to handle position %d for elemnent %s",
                                                counter, token);
200                                         break;
201                                 } // switch
202                                 tosearch = nextStart;
203                                 ++counter;
204                         } // while
205                 } // if
206         }
207
208         DEBUG_TRACE("day: %i month: %i year: %i hour: %i minute: %i second: %i latitude: %d longitude:
                %d heading: %d height: %d speed: %d", gid->day, gid->month, gid->year, gid->utc_hour,
              gid->utc_minute, gid->utc_second, gid->latitude, gid->longitude, gid->heading, gid->
              height, gid->speed );
209 }
```

Der in der Methode *setTimer* implementierte Timer wird zur Steuerung des Lesens der GPS-Daten aus der Gerätedatei (/dev/ttyACM0) des USB-GPS-Empfängers verwendet.

```
1
2  static void setTimer() {
3          struct itimerval readTimer;
4          readTimer.it_interval.tv_sec = 0; //Reset valuegetGpsAll
5          readTimer.it_interval.tv_usec = (suseconds_t) ((1.0
6                          / (double) UPDATE_FREQ_HZ) * 1000000); //Reset value
7          readTimer.it_value.tv_sec = readTimer.it_interval.tv_sec;
8          readTimer.it_value.tv_usec = readTimer.it_interval.tv_usec;
9          if (setitimer(ITIMER_REAL, &readTimer, NULL) < 0) {
10                 perror("failed to set read timer");
11         }
12 }
```

Von der Methode *readHandler* aus wird der ganze Prozess des GPS-Datenempfangs gesteuert. Anfangs werden die NMEA-0183-Datensätze von der seriellen Schnittstelle, bzw. der Gerätedatei, eingelesen und das Zerlegen dieser angestartet. Anschließend wird die mit den benötigten GPS-Daten gefüllte Struktur *GpsInfoData* in den GPS-Datencontainer geschrieben. Eine Benachrichtigung an alle anderen Komponenten wird über das Framework-Messaging-System versendet, wobei diese Message als Parame-

ter bereits den aktuellen Longitude und Latitude enthält. Zusätzlich wird das Senden der Daten an die GADA-Schnittstelle initiiert. *Anmerkung: Ein Teil des Codes ist hier auskommentiert, dieser muss nach entgültiger Zusammenführung der Komponente in das Framework noch integriert werden.*

```
1
2  void readHandler(int sig) {
3
4          if (CGPS_Component::mGPS <= 0) {
5                  return;
6          }
7
8          size_t bytesInBuffer = 0;
9          size_t bytesToRead = 0;
10         static char serialBuffer[1000];
11
12         ioctl(CGPS_Component::mGPS, FIONREAD, &bytesInBuffer);
13         if (bytesInBuffer == 0) {
14                 return;
15         } else if (bytesInBuffer > sizeof(serialBuffer)) {
16                 bytesToRead = sizeof(serialBuffer);
17         } else {
18                 bytesToRead = bytesInBuffer;
19         }
20         size_t bytesRead = read(CGPS_Component::mGPS, serialBuffer, bytesToRead);
21         if (bytesRead <= 0) {
22                 perror("read less bytes than expected");
23         } else {
24
25                 DEBUG_TRACE("bytesRead = %d", bytesRead);
26
27                 static GpsInfoData gid; // struct for all GPS-Infos
28                 // call and fill-up struct
29                 parseGPS(serialBuffer, &gid);
30
31                 // set GPS-Info-struct into GPS-DC
32                 /*
33                  CContext::getNaviGpsDataContainer()->setGpsInfoData(gid);
34
35                  // send Event, new GPS-Infos are available in DC
36                  /**   NAVI_POS,                // informs about new gps position
37                  // param1 : 1 = location | 2 = display position
38                  // param2 : latitude
39                  // param3 : longitute
40                  */
41
42                 /*
43                  bool edgeFound = CContext::getNaviDataContainer()->getCartoMapMetaData()->
                        getEdgeForGpsPosition (&edgeId, gid.longitude / 10, gid.latitude / 10);
44
45
46                  if (edgeFound)
47                  {
48                  CContext::getNaviDataContainer()->setStartEdgeID (edgeId);
49                  printf ("setting start edge to: %d\n", edgeId);
50                  }
51
52                  CNaviDispatcherComponent::sendMessage(NaviGPSID, NAVI_POS, HMIID, 1, gid.latitude,
                        gid.longitude);
53                  CNaviDispatcherComponent::sendMessage(NaviGPSID, NAVI_POS, NaviMapControlID, 1, gid.
                        latitude, gid.longitude);
54
55                  // send GPS-Data to GADA
56                  sendToGada();
57                  */
58
```

```
59          }
60          //printf("++++++ %s leave+++++++\n", __FUNCTION__);
61  }
```

B.2.2 CMiniCommander

Im CMiniCommander werden die von der seriellen Schnittstelle empfangen Kommandos interpretiert und über das Messaging-System des Frameworks an die HMI versendet.
Die Methode *recieve()* ist für das Lesen der Kommandos von der seriellen Schnittstelle zustüdig. Wird ein gültiges Kommando erkannt, so wird dieses in der Methode *interpretTelegram()* ausgewertet.

```
1   void CMiniCommander::recieve() {
2           DEBUG_TRACE("Enter");
3                   char data;
4                   int numChars = 0;
5                   int numCharsOld = 0;
6                   char stxCount = 0;
7                   char etxCount = 0;
8                   int i;
9
10                  mRecBlock.reset();
11
12                  DEBUG_TRACE("Start reading")
13                  while (1) {
14                          ioctl(mFD, FIONREAD, (int) &numChars); // read number of chars in sio buffer
15                          if (numChars != numCharsOld) {
16                                  numCharsOld = numChars;
17                          }
18
19                          // Read data from SIO buffer
20                          if (numChars != 0) {
21                                  for (i = 0; i < numChars; i++) {
22                                          read(mFD, &data, 1);
23                                          if (data == STX) // Count Start/ENd packts to ensure we have
                                                complete packets
24                                                  stxCount++;
25                                          if (data == ETX) {
26                                                  etxCount++;
27                                                  if (stxCount < etxCount) {
28                                                          DEBUG_TRACE("Bad Packet!")
29                                                          // We have a packet without start. Probably we
                                                                did not
30                                                          // receive the beginning of the packet
31                                                          // Throw it away
32                                                          mRecBlock.idxRead = mRecBlock.idxWrite + 1;
33                                                  }
34                                          }
35                                          mRecBlock.add(data); // add data to receive block
36                                  }
37
38                                  if (etxCount > 0) {
39                                          if (stxCount > 0) {
40                                                  interpretTelegram();
41                                                  etxCount--;
42                                                  stxCount--;
43                                          } else {
44                                                  // We have half a packet. Throw it away
45                                          }
46                                  }
47                          }
48                  }
49                  DEBUG_TRACE("Leave");
50  }
```

Die Methode *interpretTelegram()* ist für das Interpretieren der Kommandos verantwortlich. Wurde ein Kommando erfolgreich ausgewertet, wird das Kommando von der Methode *sendKey()* an die HMI weitergesendet.

```
/**
 *  Interpret received telegrams
 */
void CMiniCommander::interpretTelegram() {
        short numChars; // number of characters in receive buffer
        CMiniCommanderDataBlock telegram;
        char *ptr;
        char wheelCount;

        ptr = (char*) &telegram;

        // determine number of characters
        numChars = mRecBlock.count(); // number of characters in the receive buffer


        // if pointer on different ends, then adjust by use of buffer length
        if (numChars < 0) {
                numChars += MCMD_BUF_LEN;
        }

        // validate  size of characters against data block structure
        if ((unsigned short)numChars < sizeof(CMiniCommanderDataBlock)) {
                // There is no complete packet in the receive buffer.
                printf("Error imcomplete buffer in InterpretTelegem\n");
                return;
        }

        // Copy to local buffer
        for (unsigned int i = 0; i < sizeof(CMiniCommanderDataBlock); i++) {
                ptr[i] = mRecBlock.getByte();
        }

        DEBUG_TRACE("Interpreting Telegram STX=%2.0X LEN=%2x%2X SRC=%2x%2x CMD=%2X%2X DATA=%2X%2X CS
            =%2X%2X ETX=%2X",
                        telegram.stx, telegram.lengthH, telegram.lengthL,
                        telegram.sourceH, telegram.sourceL, telegram.cmdH,
                        telegram.cmdL, telegram.dataH, telegram.dataL, telegram.csH,
                        telegram.csL, telegram.etx);

        switch (telegram.cmdH) {
        case 0x35: // Key Scan or Commander present
                if (telegram.cmdL == 0x35) // Key Scan
                {
                        // Engineering Mode
                        if ((telegram.dataL & 0x06) == 0x06) // Key C+ E pressed
                                key35State = getTimestamp();
                        else if (key35State != 0) {
                                sendKey(0xf0ca); //F13
                                keyCState = 0;
                                keyEState = 0;
                                key35State = 0;
                        }

                        if (telegram.dataH & 0x02) // Key A pressed
                        {
                                keyAState = getTimestamp();
                        } else if (keyAState != 0) {
                                if (getTimestamp() - keyAState < mLengthLongKeyPressed)
                                        sendKey(0xf0be); //F1
                                else
                                        sendKey(0xf0c4); //F7
                                keyAState = 0;
                        }
```

```
63
64                      if (telegram.dataL & 0x08) // Key B pressed
65                      {
66                              keyBState = getTimestamp();
67                      } else if (keyBState != 0) {
68                              if (getTimestamp() - keyBState < mLengthLongKeyPressed)
69                                      sendKey(0xf0bf); //F2
70                              else
71                                      sendKey(0xf0c5); //F8
72                              keyBState = 0;
73                      }
74
75                      if (telegram.dataL & 0x02) // Key C pressed
76                              keyCState = getTimestamp();
77                      else if (keyCState != 0) {
78                              if (getTimestamp() - keyCState < mLengthLongKeyPressed)
79                                      sendKey(0xf0c0); //F3
80                              else
81                                      sendKey(0xf0c6); //F9
82                              keyCState = 0;
83                      }
84
85                      if (telegram.dataL & 0x01) // Key D pressed
86                              keyDState = getTimestamp();
87                      else if (keyDState != 0) {
88                              if (getTimestamp() - keyDState < mLengthLongKeyPressed)
89                                      sendKey(0xf0c1); //F4
90                              else
91                                      sendKey(0xf0c7); //F10
92                              keyDState = 0;
93                      }
94
95                      if (telegram.dataL & 0x04) // Key E pressed
96                              keyEState = getTimestamp();
97                      else if (keyEState != 0) {
98                              if (getTimestamp() - keyEState < mLengthLongKeyPressed)
99                                      sendKey(0xf0c2); //F5
100                             else
101                                     sendKey(0xf0c8); //F11
102                             keyEState = 0;
103                     }
104
105                     if (telegram.dataH & 0x01) // Key F pressed
106                             keyFState = getTimestamp();
107                     else if (keyFState != 0) {
108                             if (getTimestamp() - keyFState < mLengthLongKeyPressed)
109                                     sendKey(0xf0c3); //F6
110                             else
111                                     sendKey(0xf0c9); //F12
112                             keyFState = 0;
113                     }
114
115                     if (telegram.dataH & 0x04) // Wheel pressed
116                     {
117                             keyIncrState = getTimestamp();
118                             volumeThreshold = mVolumeThreshold;
119                     } else if (keyIncrState != 0) {
120                             if (volumeChanged == 0)
121                                     if (getTimestamp() - keyIncrState < mLengthLongKeyPressed)
122                                             sendKey(0xf00d); // Enter
123                                     else
124                                             sendKey(0xf01b); //ESC
125                             else
126                                     volumeChanged = 0; // on release rest flag
127
128                             keyIncrState = 0;
129                     }
```

```
                }
                if (telegram.cmdL == 0x41) {
                        // Commander present
                        printf("Commander Present\n");
                }

                break;
        case 0x41: // Wheel right (wrong in ISZ documentation, it says left)
                wheelCount = asciiToHex(telegram.dataL, telegram.dataH);
                if (keyIncrState != 0) // Wheel pressed
                {
                        volumeThreshold -= wheelCount;
                        if (volumeThreshold <= 0) {
                                volumeChanged = 1;
                                while (wheelCount) {
                                        sendKey(0xf054);
                                        wheelCount--;
                                }
                        }
                } else {
                        while (wheelCount) {
                                sendKey(0xf051);
                                wheelCount--;
                        }
                }
                break;
        case 0x39: // Wheel left (wrong in ISZ documentation, it says right)
                wheelCount = asciiToHex(telegram.dataL, telegram.dataH);
                if (keyIncrState != 0) // Wheel pressed
                {
                        volumeThreshold -= wheelCount;
                        if (volumeThreshold <= 0) {
                                volumeChanged = 1;
                                while (wheelCount) {
                                        sendKey(0xf052);
                                        wheelCount--;
                                }
                        }
                } else {
                        while (wheelCount) {
                                sendKey(0xf053);
                                wheelCount--;
                        }
                }
                break;
        default: // Unknown command
                DEBUG_TRACE("Unknown command received")
                break;

        }
}
```

Die Methode *sendKey(long keyValue)* schickt das ausgewertete Kommando an die HMI weiter.

```
void CMiniCommander::sendKey(long keyValue) {
        DEBUG_TRACE("value = [%ld]", keyValue)

        CMessage msg;
        msg.setReceiverID(ControllerID); // TODO: substitute controller with HMI here
        msg.setSenderID(MinicomID);
        msg.setOpcode(keyValue);
        CContext::sendMessage(msg);
}
```

B.2.3 SystemBuild

Im Header der Datei werden die Quellen der Sourcen und die Ordner definiert.
Der Befehl *make prepare* läd die Sourcen aus dem Internet herunter und kopiert die in *configuration* gespeicherten Konfigurationsdateien in die Ordner der jeweiligen Sourcen.
Zum Herunterladen und Entpacken wird das externe Script *fetch_tarball.sh* (s.u.) zur Hilfe genommen.
Der Befehl *make image* compiliert den Kernel und die Busybox.
Mit *make xserver* wird das Script *buildxserver.sh* aufgerufen, welches alle benötigten Programme installiert, die Sourcen des XServers herunterläd und compiliert.
Der Befehl *make virtual* startet eine virtuelle Maschine zum Ausprobieren des compilierten Images.
Der Befehl *make install* installiert das Image an einem beliebigen Ort.

```
1
2  ########################################
3  # file : System Build's Makefile     #
4  # desc : builds a core linux system  #
5  ########################################
6
7  ####### build variables
8
9  CURDIR                          = $(shell pwd)
10 CPUCORES                        = $(shell cat /proc/cpuinfo | grep '^processor' | wc -l)
11
12 ####### user defined variables
13 KERNEL_URL              = http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.32.tar.bz2
14 KERNEL_CONFIG           = $(CURDIR)/configuration/kernel_2_6_32.config
15
16 BUSYBOX_URL             = http://busybox.net/downloads/busybox-1.15.2.tar.bz2
17 BUSYBOX_CONFIG          = $(CURDIR)/configuration/busybox_1_15_2.config
18
19 BUILD_PARALLELJOBS      = $(shell echo $(CPUCORES)*2 | bc)
20
21 ####### calculated variables
22 BUILD_WORKDIR   = $(CURDIR)/source
23 BUILD_TARGETDIR = $(CURDIR)/image
24
25 KERNEL_DIR      = $(shell basename ${KERNEL_URL} | sed s/.tar.bz2//g)
26 KERNEL_SOURCES  = $(BUILD_WORKDIR)/$(KERNEL_DIR)
27
28 BUSYBOX_DIR     = $(shell basename ${BUSYBOX_URL} | sed s/.tar.bz2//g)
29 BUSYBOX_SOURCES = $(BUILD_WORKDIR)/$(BUSYBOX_DIR)
30
31 CPIO_GEN        = $(KERNEL_SOURCES)/usr/gen_init_cpio
32 TOOLS_DIR       = $(CURDIR)/tools
33 MODULE_LIST     = $(shell find ${BUILD_TARGETDIR}/modules -name '*.ko')
34
35 ####### build targets
36
37 all: force
38         @echo ""
39         @echo "              ++++++ SYSTEM BUILD MAKE TARGETS ++++++"
40         @echo ""
41         @echo "   prepare ..............: download, extract and configure necessary software"
42         @echo "   image ................: builds a system image"
43         @echo "   xserver ..............: builds the xserver using the separate build script"
44         @echo "   virtual ..............: starts the image within qemu virtualization solution"
45         @echo "   install ..............: installs necessary files for startup to a given dir"
46         @echo ""
47         @echo "   variables:"
48         @echo "       working dir .......: $(CURDIR)"
49         @echo "       cpu cores .........: $(CPUCORES)"
50         @echo "       parallel jobs .....: $(BUILD_PARALLELJOBS)"
```

```
51          @echo ""
52
53
54  ## download and configuration of sources
55
56  fetchandconfigkernel: force
57          @echo "download and extract sources of linux kernel"
58          cd $(BUILD_WORKDIR); $(TOOLS_DIR)/fetch_tarball.sh $(KERNEL_URL)
59          @echo "configure linux kernel"
60          cp $(KERNEL_CONFIG) $(KERNEL_SOURCES)/.config
61          cd $(KERNEL_SOURCES); make oldconfig
62
63  fetchandconfigbusybox: force
64          @echo "download and extract sources of busybox"
65          cd $(BUILD_WORKDIR); $(TOOLS_DIR)/fetch_tarball.sh $(BUSYBOX_URL)
66          @echo "configure busybox"
67          cp $(BUSYBOX_CONFIG) $(BUSYBOX_SOURCES)/.config
68          cd $(BUSYBOX_SOURCES); make oldconfig
69
70  prepare: fetchandconfigkernel fetchandconfigbusybox
71
72  image: kernel busybox initramfs
73
74  clean:
75          @echo ">>> I will REMOVE ALL generated data!!"
76          @echo "    Press ^C now if you do NOT want to do this."
77          @read IGNORE
78          @echo ">>> now clean ..."
79          rm -rf $(BUILD_WORKDIR)/*
80          rm -rf $(BUILD_TARGETDIR)/*
81          @echo ">>> done"
82
83  initramfs: force
84          cp initramfs/initramfs.conf initramfs/initramfs.tmp
85          chmod +w initramfs/initramfs.tmp
86          @for i in $(MODULE_LIST); do \
87          echo "file /modules/`basename $$i` $$i 755 0 0" >> initramfs/initramfs.tmp; done
88          $(CPIO_GEN) initramfs/initramfs.tmp > image/initramfs.cpio
89          rm initramfs/initramfs.tmp
90          @du -h image/initramfs.cpio
91
92  virtual: force
93           qemu -kernel $(BUILD_TARGETDIR)/bzImage -initrd $(BUILD_TARGETDIR)/initramfs.cpio /dev/null
94
95  kernel: force
96          cd $(KERNEL_SOURCES); make -j$(BUILD_PARALLELJOBS)
97          cp $(KERNEL_SOURCES)/arch/x86/boot/bzImage $(BUILD_TARGETDIR)/
98          rm -rf $(CURDIR)/modules
99          mkdir -p $(CURDIR)/modules
100         cd $(KERNEL_SOURCES); INSTALL_MOD_PATH=$(BUILD_TARGETDIR)/modules make modules_install
101
102 busybox: force
103         cd $(BUSYBOX_SOURCES); make -j$(BUILD_PARALLELJOBS)
104         cp -p $(BUSYBOX_SOURCES)/busybox $(BUILD_TARGETDIR)/
105
106 install: force
107         @if [ -n "$(TARGET)" ] && [ -d "$(TARGET)" ]; then \
108                 cp -vp $(BUILD_TARGETDIR)/bzImage $(TARGET)/ ; \
109                 cp -vp $(BUILD_TARGETDIR)/initramfs.cpio $(TARGET)/ ; \
110         else \
111                 echo "ERROR: no valid target defined ( append 'TARGET=<dir>' )" ; \
112         fi
113
114 xserver: force
115         ./buildxserver.sh
116
117 force: ;
```

```
118
119 ## EOF
```

Damit die Makedatei etwas übersichtlicher ist, wurde die Funktionaltität zum Herunterladen und Entpacken der Sourcen in ein weiteres Shell-Script ausgelagert. Dieses befindet sich im Ordner *tool* und heisst *fetch_tarball.sh*. Als Übergabeparameter wird hier die URL des tar-Archives erwartet. Mit dem Kommando *wget* wird das übergebene Archiv heruntergeladen und anschließend mit dem Befehl *tar -xf* entpackt.

```
1
2  #!/bin/sh
3
4  ## fetch_tarball.sh
5  ##
6  ## downloads and extracts a given tarball
7  ## specified by its url
8  ##
9  ## as target directory the name of the tarball
10 ## without its suffix is used, located
11 ## within the current directory
12 ##
13 ## only tarballs with suffix '.tar.bz2' are
14 ## supported
15
16 URL="$1"
17
18 if [ ! -n "$URL" ] ; then
19         echo "ERROR: no url specified"
20         exit 1
21 fi
22
23
24 DIR=`basename ${URL} | sed s/.tar.bz2//g`
25
26 if [ ! -r `basename ${URL}` ] ; then
27         wget $1
28 else
29         echo ">>> using already available tarball `basename ${URL}`"
30 fi
31
32 if [ ! -r `basename ${URL}` ] ; then
33         echo "ERROR: failed to access tarball"
34         exit 2
35 else
36         if [ ! -d "$DIR" ] ; then
37                 echo ">>> extracting `basename ${URL}`"
38                 tar -xf `basename ${URL}`
39         else
40                 echo ">>> using already available directory [$DIR]"
41         fi
42 fi
43
44 exit 0
45
46 ## EOF
```

Die Datei *initramfs.conf* ist für das Erstellen des Dateisystems im Image zuständig. Hier müssen alle Ordner und Dateien angegeben werden, die zur Startzeit verfügbar sein sollen. Zu Beginn jeder Zeile steht um welchen Typ es sich handelt (*dir* für Directory, *nod* für Device Nodes, *file* für File und *slink* für einen Symbolic-Link). Anschließend folgt der Bezeichner im zukünftigen Dateisystem. Dann, falls es ein Datei ist, wo sich das Datei aktuell befindet, bzw. wenn es ein Symbolic-Link ist, wohin dieser verweist und zu guter letzt die Dateirechte.

Der Kernel ist so konfiguriert, dass beim Starten automatisch alle sich in Ordner */init* befindlichen Scripte ausgeführt werden.
Mit dem Eintrag *file /init ./initramfs/init.sh 755 0 0* wird angegeben, dass beim Starten des Kernels die im Ordner */initramfs* liegende Datei *init.sh* ausgeführt werden soll.

```
## initramfs.conf
##
## configuration file to be used with
## the application 'gen_init_cpio', which
## can be found within the kernel sources
## <kernelsrc>/usr/gen_init_cpio
##
## it defines the structure and content
## of a initial ram filesystem used during
## the systems startup
##
## for more information see the kernels
## documentation folder
## <kernelsrc>/Documentation/filesystems/ramfs-rootfs-initramfs.txt

## directory structure
dir /bin 755 1000 1000
dir /dev 755 0 0
dir /etc 755 0 0
dir /lib 755 0 0
dir /proc 755 0 0
dir /sbin 755 0 0
dir /sys 755 0 0
dir /tmp 777 0 0
dir /usr 755 0 0
dir /usr/bin 755 0 0
dir /usr/sbin 755 0 0
dir /usr/local 755 0 0
dir /dev/shm/ 755 0 0
dir /dev/input/ 755 0 0
dir /dev/mqueue 755 0 0
dir /modules 755 0 0
dir /mnt/ 755 0 0
dir /mnt/sda1 755 0 0
dir /mnt/sda2 755 0 0

## device nodes
nod /dev/mem 640 0 0 c 1 1
nod /dev/kmem 640 0 0 c 1 2
nod /dev/null 640 0 0 c 1 3
nod /dev/zero 640 0 0 c 1 5
nod /dev/random 640 0 0 c 1 8
nod /dev/urandom 640 0 0 c 1 9
nod /dev/tty 666 0 0 c 5 0
nod /dev/tty0 666 0 0 c 4 0
nod /dev/tty1 666 0 0 c 4 1
nod /dev/tty2 666 0 0 c 4 2
nod /dev/ttyS0 666 0 0 c 4 64
nod /dev/console 640 0 0 c 5 1
nod /dev/ram 640 0 0 b 1 1
nod /dev/ram0 640 0 0 b 1 0
nod /dev/loop 640 0 0 b 7 0
nod /dev/psaux 640 0 0 c 10 1
nod /dev/fb0 640 0 0 c 29 0
nod /dev/ttyUSB0 666 0 0 c 188 0
nod /dev/ttyUSB1 666 0 0 c 188 1
nod /dev/ttyACM0 666 0 0 c 166 0
nod /dev/ttyACM1 666 0 0 c 166 1
nod /dev/hda 666 0 0 b 3 0
nod /dev/hda1 666 0 0 b 3 1
```

```
nod /dev/hda2 666 0 0 b 3 2
nod /dev/sda 666 0 0 b 8 0
nod /dev/sda1 666 0 0 b 8 1
nod /dev/sda2 666 0 0 b 8 2
nod /dev/sda3 666 0 0 b 8 3
nod /dev/sda4 666 0 0 b 8 4
nod /dev/sda5 666 0 0 b 8 5
nod /dev/sda6 666 0 0 b 8 6
nod /dev/sda7 666 0 0 b 8 7
nod /dev/sda8 666 0 0 b 8 8

nod /dev/input/mice 640 0 0 c 13 63
nod /dev/input/mouse0 640 0 0 c 13 32
nod /dev/input/mouse1 640 0 0 c 13 33

## system libs

file  /lib/ld-2.10.1.so          ./lib/ld-2.10.1.so              755 0 0
slink /lib/ld-linux.so.2         /lib/ld-2.10.1.so               755 0 0
file  /lib/libc-2.10.1.so        ./lib/libc-2.10.1.so            755 0 0
slink /lib/libc.so.6             /lib/libc-2.10.1.so             755 0 0
file  /lib/libm-2.10.1.so        ./lib/libm-2.10.1.so            755 0 0
slink /lib/libm.so.6             /lib/libm-2.10.1.so             755 0 0
file  /lib/librt-2.10.1.so       ./lib/librt-2.10.1.so           755 0 0
slink /lib/librt.so.1            /lib/librt-2.10.1.so            755 0 0
file  /lib/libpthread-2.10.1.so  ./lib/libpthread-2.10.1.so      755 0 0
slink /lib/libpthread.so.0       /lib/libpthread-2.10.1.so       755 0 0
file  /lib/libstdc++.so.6.0.13   ./lib/libstdc++.so.6.0.13       755 0 0
slink /lib/libstdc++.so.6        /lib/libstdc++.so.6.0.13        755 0 0
file  /lib/libgcc_s.so.1         ./lib/libgcc_s.so.1             755 0 0


##file  /lib/linux-gate.so.1            ./lib/linux-gate.so.1   755 0 0
file  /lib/libglut.so.3               ./lib/libglut.so.3      755 0 0
file  /lib/libmysqlclient.so.15       ./lib/libmysqlclient.so.15      755 0 0
file  /lib/libSDL_image-1.2.so.0      ./lib/libSDL_image-1.2.so.0     755 0 0
file  /lib/libSDL-1.2.so.0            ./lib/libSDL-1.2.so.0   755 0 0
file  /lib/libGL.so.1                 ./lib/libGL.so.1        755 0 0
file  /lib/libstdc++.so.6             ./lib/libstdc++.so.6    755 0 0
file  /lib/libgcc_s.so.1              ./lib/libgcc_s.so.1     755 0 0
file  /lib/libc.so.6                  ./lib/libc.so.6         755 0 0
file  /lib/libGLU.so.1                ./lib/libGLU.so.1       755 0 0
file  /lib/libXext.so.6               ./lib/libXext.so.6      755 0 0
file  /lib/libX11.so.6                ./lib/libX11.so.6       755 0 0
file  /lib/libXxf86vm.so.1            ./lib/libXxf86vm.so.1   755 0 0
file  /lib/libXi.so.6                 ./lib/libXi.so.6        755 0 0
file  /lib/libcrypt.so.1              ./lib/libcrypt.so.1     755 0 0
file  /lib/libnsl.so.1                ./lib/libnsl.so.1       755 0 0
file  /lib/libz.so.1                  ./lib/libz.so.1         755 0 0
file  /lib/libpng12.so.0              ./lib/libpng12.so.0     755 0 0
file  /lib/libjpeg.so.62              ./lib/libjpeg.so.62     755 0 0
file  /lib/libtiff.so.4               ./lib/libtiff.so.4      755 0 0
file  /lib/libasound.so.2             ./lib/libasound.so.2    755 0 0
file  /lib/libdl.so.2                 ./lib/libdl.so.2        755 0 0
file  /lib/libdirectfb-1.2.so.0       ./lib/libdirectfb-1.2.so.0      755 0 0
file  /lib/libfusion-1.2.so.0         ./lib/libfusion-1.2.so.0        755 0 0
file  /lib/libdirect-1.2.so.0         ./lib/libdirect-1.2.so.0        755 0 0
file  /lib/libXdamage.so.1            ./lib/libXdamage.so.1   755 0 0
file  /lib/libXfixes.so.3             ./lib/libXfixes.so.3    755 0 0
file  /lib/libdrm.so.2                ./lib/libdrm.so.2       755 0 0
file  /lib/ld-linux.so.2              ./lib/ld-linux.so.2     755 0 0
file  /lib/libXau.so.6                ./lib/libXau.so.6       755 0 0
file  /lib/libxcb.so.1                ./lib/libxcb.so.1       755 0 0
file  /lib/librt.so.1                 ./lib/librt.so.1        755 0 0
file  /lib/libXdmcp.so.6              ./lib/libXdmcp.so.6     755 0 0

file  /lib/libcrypto.so.0.9.8         ./lib/libcrypto.so.0.9.8        755 0 0
```

```
file  /lib/libpciaccess.so.0            ./lib/libpciaccess.so.0         755 0 0
file  /lib/libXfont.so.1                ./lib/libXfont.so.1             755 0 0
file  /lib/libpixman-1.so.0             ./lib/libpixman-1.so.0          755 0 0
file  /lib/libfreetype.so.6             ./lib/libfreetype.so.6          755 0 0
file  /lib/libfontenc.so.1              ./lib/libfontenc.so.1           755 0 0

file  /lib/libgcrypt.so.11.5.2          ./lib/libgcrypt.so.11.5.2       755 0 0
slink /lib/libgcrypt.so.11              /lib/libgcrypt.so.11.5.2        755 0 0
file  /lib/libgpg-error.so.0.4.0        ./lib/libgpg-error.so.0.4.0     755 0 0
slink /lib/libgpg-error.so.0            /lib/libgpg-error.so.0.4.0      755 0 0

## the init script
file /init ./initramfs/init.sh 755 0 0

## busybox multiwrapper application
file /bin/busybox ./image/busybox 755 0 0

## xserver
#file /bin/xinit ./image/xinit 755 0 0
#file /bin/Xorg ./image/Xorg 755 0 0
#file /bin/xinit ./image/xinit 755 0 0
#file /bin/Xorg ./image/Xorg 755 0 0
#slink /bin/X   /bin/Xorg 755 0 0
#file /bin/Xvfb ./image/Xvfb 755 0 0
#file /bin/Xnest ./image/Xnest 755 0 0

file  /bin/startx ./configuration/startx        755 0 0
file  /bin/xinit ./image/xserver/bin/xinit 755 0 0
file  /bin/Xorg ./image/xserver/bin/Xorg 755 0 0
slink /bin/X /bin/Xorg 755 0 0

dir /lib/X11                            755 0 0
dir /lib/X11/xinit                      755 0 0
dir /lib/xorg                           755 0 0
dir /lib/xorg/modules                   755 0 0
dir /lib/xorg/modules/multimedia        755 0 0
dir /lib/xorg/modules/linux             755 0 0
dir /lib/xorg/modules/extensions        755 0 0
dir /lib/xorg/modules/input             755 0 0
dir /lib/xorg/modules/drivers           755 0 0

## xserver modules

file /lib/swrast_dri.so ./image/xserver/mesa/mesa/lib/swrast_dri.so 755 0 0

#file /lib/xorg/modules/multimedia/msp3430_drv.so ./image/xserver/lib/xorg/modules/multimedia/
    msp3430_drv.so  755 0 0
#file /lib/xorg/modules/multimedia/tda8425_drv.so ./image/xserver/lib/xorg/modules/multimedia/
    tda8425_drv.so  755 0 0
#file /lib/xorg/modules/multimedia/tda9850_drv.so ./image/xserver/lib/xorg/modules/multimedia/
    tda9850_drv.so  755 0 0
#file /lib/xorg/modules/multimedia/fi1236_drv.so ./image/xserver/lib/xorg/modules/multimedia/
    fi1236_drv.so  755 0 0
#file /lib/xorg/modules/multimedia/tda9885_drv.so ./image/xserver/lib/xorg/modules/multimedia/
    tda9885_drv.so  755 0 0
#file /lib/xorg/modules/multimedia/bt829_drv.so ./image/xserver/lib/xorg/modules/multimedia/bt829_drv.
    so  755 0 0
#file /lib/xorg/modules/multimedia/theatre_drv.so ./image/xserver/lib/xorg/modules/multimedia/
    theatre_drv.so  755 0 0
#file /lib/xorg/modules/multimedia/theatre_detect_drv.so ./image/xserver/lib/xorg/modules/multimedia/
    theatre_detect_drv.so  755 0 0
#file /lib/xorg/modules/multimedia/uda1380_drv.so ./image/xserver/lib/xorg/modules/multimedia/
    uda1380_drv.so  755 0 0
#file /lib/xorg/modules/multimedia/theatre200_drv.so ./image/xserver/lib/xorg/modules/multimedia/
    theatre200_drv.so  755 0 0

```

```
184 file /lib/xorg/modules/linux/libfbdevhw.so ./image/xserver/lib/xorg/modules/linux/libfbdevhw.so 755 0
      0
185 file /lib/xorg/modules/extensions/libdbe.so ./image/xserver/lib/xorg/modules/extensions/libdbe.so 755
      0 0
186 file /lib/xorg/modules/extensions/libdri.so ./image/xserver/lib/xorg/modules/extensions/libdri.so 755
      0 0
187 file /lib/xorg/modules/extensions/libglx.so ./image/xserver/lib/xorg/modules/extensions/libglx.so 755
      0 0
188 file /lib/xorg/modules/extensions/libextmod.so ./image/xserver/lib/xorg/modules/extensions/libextmod.
    so 755 0 0
189 file /lib/xorg/modules/extensions/libdri2.so ./image/xserver/lib/xorg/modules/extensions/libdri2.so
    755 0 0
190 file /lib/xorg/modules/libvgahw.so ./image/xserver/lib/xorg/modules/libvgahw.so 755 0 0
191
192 file /lib/xorg/modules/input/vmmouse_drv.so ./image/xserver/lib/xorg/modules/input/vmmouse_drv.so 755
      0 0
193 #file /lib/xorg/modules/input/evdev_drv.so ./image/xserver/lib/xorg/modules/input/evdev_drv.so 755 0
    0
194 #file /lib/xorg/modules/input/aiptek_drv.so ./image/xserver/lib/xorg/modules/input/aiptek_drv.so 755
    0 0
195 file /lib/xorg/modules/input/mouse_drv.so ./image/xserver/lib/xorg/modules/input/mouse_drv.so 755 0 0
196 #file /lib/xorg/modules/input/acecad_drv.so ./image/xserver/lib/xorg/modules/input/acecad_drv.so 755
    0 0
197 #file /lib/xorg/modules/input/void_drv.so ./image/xserver/lib/xorg/modules/input/void_drv.so 755 0 0
198 #file /lib/xorg/modules/input/joystick_drv.so ./image/xserver/lib/xorg/modules/input/joystick_drv.so
    755 0 0
199 file /lib/xorg/modules/input/kbd_drv.so ./image/xserver/lib/xorg/modules/input/kbd_drv.so 755 0 0
200 file /lib/xorg/modules/input/synaptics_drv.so ./image/xserver/lib/xorg/modules/input/synaptics_drv.so
      755 0 0
201
202 file /lib/xorg/modules/libshadow.so ./image/xserver/lib/xorg/modules/libshadow.so 755 0 0
203 file /lib/xorg/modules/libfb.so ./image/xserver/lib/xorg/modules/libfb.so 755 0 0
204 file /lib/xorg/modules/libxaa.so ./image/xserver/lib/xorg/modules/libxaa.so 755 0 0
205 file /lib/xorg/modules/libshadowfb.so ./image/xserver/lib/xorg/modules/libshadowfb.so 755 0 0
206 file /lib/xorg/modules/libxf8_16bpp.so ./image/xserver/lib/xorg/modules/libxf8_16bpp.so 755 0 0
207
208 file /lib/xorg/modules/drivers/cirrus_drv.so ./image/xserver/lib/xorg/modules/drivers/cirrus_drv.so
    755 0 0
209 #file /lib/xorg/modules/drivers/ztv_drv.so ./image/xserver/lib/xorg/modules/drivers/ztv_drv.so 755 0
    0
210 #file /lib/xorg/modules/drivers/tseng_drv.so ./image/xserver/lib/xorg/modules/drivers/tseng_drv.so
    755 0 0
211 #file /lib/xorg/modules/drivers/sis_drv.so ./image/xserver/lib/xorg/modules/drivers/sis_drv.so 755 0
    0
212 #file /lib/xorg/modules/drivers/v4l_drv.so ./image/xserver/lib/xorg/modules/drivers/v4l_drv.so 755 0
    0
213 #file /lib/xorg/modules/drivers/r128_drv.so ./image/xserver/lib/xorg/modules/drivers/r128_drv.so 755
    0 0
214 #file /lib/xorg/modules/drivers/trident_drv.so ./image/xserver/lib/xorg/modules/drivers/trident_drv.so
      755 0 0
215 #file /lib/xorg/modules/drivers/radeon_drv.so ./image/xserver/lib/xorg/modules/drivers/radeon_drv.so
    755 0 0
216 #file /lib/xorg/modules/drivers/ark_drv.so ./image/xserver/lib/xorg/modules/drivers/ark_drv.so 755 0
    0
217 #file /lib/xorg/modules/drivers/s3virge_drv.so ./image/xserver/lib/xorg/modules/drivers/s3virge_drv.so
      755 0 0
218 #file /lib/xorg/modules/drivers/newport_drv.so ./image/xserver/lib/xorg/modules/drivers/newport_drv.so
      755 0 0
219 #file /lib/xorg/modules/drivers/rendition_drv.so ./image/xserver/lib/xorg/modules/drivers/
    rendition_drv.so 755 0 0
220 #file /lib/xorg/modules/drivers/apm_drv.so ./image/xserver/lib/xorg/modules/drivers/apm_drv.so 755 0
    0
221 #file /lib/xorg/modules/drivers/voodoo_drv.so ./image/xserver/lib/xorg/modules/drivers/voodoo_drv.so
    755 0 0
222 #file /lib/xorg/modules/drivers/mga_drv.so ./image/xserver/lib/xorg/modules/drivers/mga_drv.so 755 0
    0
```

```
#file /lib/xorg/modules/drivers/neomagic_drv.so ./image/xserver/lib/xorg/modules/drivers/neomagic_drv.so 755 0 0
file /lib/xorg/modules/drivers/intel_drv.so ./image/xserver/lib/xorg/modules/drivers/intel_drv.so 755 0 0
#file /lib/xorg/modules/drivers/geode_drv.so ./image/xserver/lib/xorg/modules/drivers/geode_drv.so 755 0 0
#file /lib/xorg/modules/drivers/i128_drv.so ./image/xserver/lib/xorg/modules/drivers/i128_drv.so 755 0 0
#file /lib/xorg/modules/drivers/cirrus_laguna.so ./image/xserver/lib/xorg/modules/drivers/cirrus_laguna.so 755 0 0
#file /lib/xorg/modules/drivers/radeonhd_drv.so ./image/xserver/lib/xorg/modules/drivers/radeonhd_drv.so 755 0 0
file /lib/xorg/modules/drivers/vesa_drv.so ./image/xserver/lib/xorg/modules/drivers/vesa_drv.so 755 0 0
file /lib/xorg/modules/drivers/xgi_drv.so ./image/xserver/lib/xorg/modules/drivers/xgi_drv.so 755 0 0
file /lib/xorg/modules/drivers/dummy_drv.so ./image/xserver/lib/xorg/modules/drivers/dummy_drv.so 755 0 0
#file /lib/xorg/modules/drivers/sunffb_drv.so ./image/xserver/lib/xorg/modules/drivers/sunffb_drv.so 755 0 0
#file /lib/xorg/modules/drivers/sisusb_drv.so ./image/xserver/lib/xorg/modules/drivers/sisusb_drv.so 755 0 0
file /lib/xorg/modules/drivers/nv_drv.so ./image/xserver/lib/xorg/modules/drivers/nv_drv.so 755 0 0
#file /lib/xorg/modules/drivers/ast_drv.so ./image/xserver/lib/xorg/modules/drivers/ast_drv.so 755 0 0
#file /lib/xorg/modules/drivers/savage_drv.so ./image/xserver/lib/xorg/modules/drivers/savage_drv.so 755 0 0
#file /lib/xorg/modules/drivers/cirrus_alpine.so ./image/xserver/lib/xorg/modules/drivers/cirrus_alpine.so 755 0 0
#file /lib/xorg/modules/drivers/mach64_drv.so ./image/xserver/lib/xorg/modules/drivers/mach64_drv.so 755 0 0
#file /lib/xorg/modules/drivers/ati_drv.so ./image/xserver/lib/xorg/modules/drivers/ati_drv.so 755 0 0
#file /lib/xorg/modules/drivers/xgixp_drv.so ./image/xserver/lib/xorg/modules/drivers/xgixp_drv.so 755 0 0
#file /lib/xorg/modules/drivers/s3_drv.so ./image/xserver/lib/xorg/modules/drivers/s3_drv.so 755 0 0
#file /lib/xorg/modules/drivers/tga_drv.so ./image/xserver/lib/xorg/modules/drivers/tga_drv.so 755 0 0
#file /lib/xorg/modules/drivers/vmware_drv.so ./image/xserver/lib/xorg/modules/drivers/vmware_drv.so 755 0 0
#file /lib/xorg/modules/drivers/chips_drv.so ./image/xserver/lib/xorg/modules/drivers/chips_drv.so 755 0 0
#file /lib/xorg/modules/drivers/tdfx_drv.so ./image/xserver/lib/xorg/modules/drivers/tdfx_drv.so 755 0 0
file /lib/xorg/modules/drivers/fbdev_drv.so ./image/xserver/lib/xorg/modules/drivers/fbdev_drv.so 755 0 0
#file /lib/xorg/modules/drivers/siliconmotion_drv.so ./image/xserver/lib/xorg/modules/drivers/siliconmotion_drv.so 755 0 0
#file /lib/xorg/modules/drivers/glint_drv.so ./image/xserver/lib/xorg/modules/drivers/glint_drv.so 755 0 0
#file /lib/xorg/modules/drivers/i740_drv.so ./image/xserver/lib/xorg/modules/drivers/i740_drv.so 755 0 0

file /lib/xorg/modules/libint10.so ./image/xserver/lib/xorg/modules/libint10.so 755 0 0
file /lib/xorg/modules/libwfb.so ./image/xserver/lib/xorg/modules/libwfb.so 755 0 0
file /lib/xorg/modules/libvbe.so ./image/xserver/lib/xorg/modules/libvbe.so 755 0 0
file /lib/xorg/modules/libexa.so ./image/xserver/lib/xorg/modules/libexa.so 755 0 0

## testapp
## file /bin/testapp ./image/testapp 755 0 0


## Navi 3D
file /bin/Navi/Navi3D_Client    ./image/Navi3D/Navi3D_Client    755 0 0
file /bin/Navi/gadata           ./image/Navi3D/gadata           755 0 0
file /bin/Navi/gagps            ./image/Navi3D/gagps            755 0 0
file /bin/Navi/bild1.bmp        ./image/Navi3D/bild1.bmp        755 0 0
file /bin/Navi/Profiles.xml     ./image/Navi3D/Profiles.xml     755 0 0
```

```
266
267
268
269 ## symbolic links for busybox environment
270 slink /usr/bin/[ /bin/busybox 755 0 0
271 slink /usr/bin/[[ /bin/busybox 755 0 0
272 slink /bin/addgroup /bin/busybox 755 0 0
273 slink /bin/adduser /bin/busybox 755 0 0
274 slink /sbin/adjtimex /bin/busybox 755 0 0
275 slink /usr/bin/ar /bin/busybox 755 0 0
276 slink /sbin/arp /bin/busybox 755 0 0
277 slink /usr/bin/arping /bin/busybox 755 0 0
278 slink /bin/ash /bin/busybox 755 0 0
279 slink /usr/bin/awk /bin/busybox 755 0 0
280 slink /usr/bin/basename /bin/busybox 755 0 0
281 slink /usr/bin/beep /bin/busybox 755 0 0
282 slink /sbin/blkid /bin/busybox 755 0 0
283 slink /usr/sbin/brctl /bin/busybox 755 0 0
284 slink /usr/bin/bunzip2 /bin/busybox 755 0 0
285 slink /usr/bin/bzcat /bin/busybox 755 0 0
286 slink /usr/bin/bzip2 /bin/busybox 755 0 0
287 slink /usr/bin/cal /bin/busybox 755 0 0
288 slink /bin/cat /bin/busybox 755 0 0
289 slink /bin/catv /bin/busybox 755 0 0
290 slink /usr/bin/chat /bin/busybox 755 0 0
291 slink /bin/chattr /bin/busybox 755 0 0
292 slink /bin/chgrp /bin/busybox 755 0 0
293 slink /bin/chmod /bin/busybox 755 0 0
294 slink /bin/chown /bin/busybox 755 0 0
295 slink /usr/sbin/chpasswd /bin/busybox 755 0 0
296 slink /usr/bin/chpst /bin/busybox 755 0 0
297 slink /usr/sbin/chroot /bin/busybox 755 0 0
298 slink /usr/bin/chrt /bin/busybox 755 0 0
299 slink /usr/bin/chvt /bin/busybox 755 0 0
300 slink /usr/bin/cksum /bin/busybox 755 0 0
301 slink /usr/bin/clear /bin/busybox 755 0 0
302 slink /usr/bin/cmp /bin/busybox 755 0 0
303 slink /usr/bin/comm /bin/busybox 755 0 0
304 slink /bin/cp /bin/busybox 755 0 0
305 slink /bin/cpio /bin/busybox 755 0 0
306 slink /usr/sbin/crond /bin/busybox 755 0 0
307 slink /usr/bin/crontab /bin/busybox 755 0 0
308 slink /usr/bin/cryptpw /bin/busybox 755 0 0
309 slink /usr/bin/cut /bin/busybox 755 0 0
310 slink /bin/date /bin/busybox 755 0 0
311 slink /usr/bin/dc /bin/busybox 755 0 0
312 slink /bin/dd /bin/busybox 755 0 0
313 slink /usr/bin/deallocvt /bin/busybox 755 0 0
314 slink /bin/delgroup /bin/busybox 755 0 0
315 slink /bin/deluser /bin/busybox 755 0 0
316 slink /sbin/depmod /bin/busybox 755 0 0
317 slink /sbin/devmem /bin/busybox 755 0 0
318 slink /bin/df /bin/busybox 755 0 0
319 slink /usr/sbin/dhcprelay /bin/busybox 755 0 0
320 slink /usr/bin/diff /bin/busybox 755 0 0
321 slink /usr/bin/dirname /bin/busybox 755 0 0
322 slink /bin/dmesg /bin/busybox 755 0 0
323 slink /usr/sbin/dnsd /bin/busybox 755 0 0
324 slink /bin/dnsdomainname /bin/busybox 755 0 0
325 slink /usr/bin/dos2unix /bin/busybox 755 0 0
326 slink /usr/bin/du /bin/busybox 755 0 0
327 slink /bin/dumpkmap /bin/busybox 755 0 0
328 slink /usr/bin/dumpleases /bin/busybox 755 0 0
329 slink /bin/echo /bin/busybox 755 0 0
330 slink /bin/ed /bin/busybox 755 0 0
331 slink /bin/egrep /bin/busybox 755 0 0
332 slink /usr/bin/eject /bin/busybox 755 0 0
```

```
slink /usr/bin/env /bin/busybox 755 0 0
slink /usr/bin/envdir /bin/busybox 755 0 0
slink /usr/bin/envuidgid /bin/busybox 755 0 0
slink /usr/bin/ether-wake /bin/busybox 755 0 0
slink /usr/bin/expand /bin/busybox 755 0 0
slink /usr/bin/expr /bin/busybox 755 0 0
slink /usr/sbin/fakeidentd /bin/busybox 755 0 0
slink /bin/false /bin/busybox 755 0 0
slink /usr/sbin/fbset /bin/busybox 755 0 0
slink /sbin/fbsplash /bin/busybox 755 0 0
slink /bin/fdflush /bin/busybox 755 0 0
slink /usr/bin/fdformat /bin/busybox 755 0 0
slink /sbin/fdisk /bin/busybox 755 0 0
slink /bin/fgrep /bin/busybox 755 0 0
slink /usr/bin/find /bin/busybox 755 0 0
slink /sbin/findfs /bin/busybox 755 0 0
slink /usr/bin/fold /bin/busybox 755 0 0
slink /usr/bin/free /bin/busybox 755 0 0
slink /sbin/freeramdisk /bin/busybox 755 0 0
slink /sbin/fsck /bin/busybox 755 0 0
slink /sbin/fsck.minix /bin/busybox 755 0 0
slink /bin/fsync /bin/busybox 755 0 0
slink /usr/sbin/ftpd /bin/busybox 755 0 0
slink /usr/bin/ftpget /bin/busybox 755 0 0
slink /usr/bin/ftpput /bin/busybox 755 0 0
slink /usr/bin/fuser /bin/busybox 755 0 0
slink /bin/getopt /bin/busybox 755 0 0
slink /sbin/getty /bin/busybox 755 0 0
slink /bin/grep /bin/busybox 755 0 0
slink /bin/gunzip /bin/busybox 755 0 0
slink /bin/gzip /bin/busybox 755 0 0
slink /sbin/halt /bin/busybox 755 0 0
slink /usr/bin/hd /bin/busybox 755 0 0
slink /sbin/hdparm /bin/busybox 755 0 0
slink /usr/bin/head /bin/busybox 755 0 0
slink /usr/bin/hexdump /bin/busybox 755 0 0
slink /usr/bin/hostid /bin/busybox 755 0 0
slink /bin/hostname /bin/busybox 755 0 0
slink /usr/sbin/httpd /bin/busybox 755 0 0
slink /bin/hush /bin/busybox 755 0 0
slink /sbin/hwclock /bin/busybox 755 0 0
slink /usr/bin/id /bin/busybox 755 0 0
slink /sbin/ifconfig /bin/busybox 755 0 0
slink /sbin/ifdown /bin/busybox 755 0 0
slink /sbin/ifenslave /bin/busybox 755 0 0
slink /usr/bin/ifplugd /bin/busybox 755 0 0
slink /sbin/ifup /bin/busybox 755 0 0
slink /usr/sbin/inetd /bin/busybox 755 0 0
slink /sbin/init /bin/busybox 755 0 0
slink /sbin/insmod /bin/busybox 755 0 0
slink /usr/bin/install /bin/busybox 755 0 0
slink /bin/ionice /bin/busybox 755 0 0
slink /bin/ip /bin/busybox 755 0 0
slink /bin/ipaddr /bin/busybox 755 0 0
slink /bin/ipcalc /bin/busybox 755 0 0
slink /usr/bin/ipcrm /bin/busybox 755 0 0
slink /usr/bin/ipcs /bin/busybox 755 0 0
slink /bin/iplink /bin/busybox 755 0 0
slink /bin/iproute /bin/busybox 755 0 0
slink /bin/iprule /bin/busybox 755 0 0
slink /bin/iptunnel /bin/busybox 755 0 0
slink /usr/bin/kbd_mode /bin/busybox 755 0 0
slink /bin/kill /bin/busybox 755 0 0
slink /usr/bin/killall /bin/busybox 755 0 0
slink /usr/bin/killall5 /bin/busybox 755 0 0
slink /sbin/klogd /bin/busybox 755 0 0
slink /usr/bin/last /bin/busybox 755 0 0
```

```
slink /usr/bin/length /bin/busybox 755 0 0
slink /usr/bin/less /bin/busybox 755 0 0
slink /bin/linux32 /bin/busybox 755 0 0
slink /bin/linux64 /bin/busybox 755 0 0
slink /linuxrc /bin/busybox 755 0 0
slink /bin/ln /bin/busybox 755 0 0
slink /usr/sbin/loadfont /bin/busybox 755 0 0
slink /sbin/loadkmap /bin/busybox 755 0 0
slink /usr/bin/logger /bin/busybox 755 0 0
slink /bin/login /bin/busybox 755 0 0
slink /usr/bin/logname /bin/busybox 755 0 0
slink /sbin/logread /bin/busybox 755 0 0
slink /sbin/losetup /bin/busybox 755 0 0
slink /usr/sbin/lpd /bin/busybox 755 0 0
slink /usr/bin/lpq /bin/busybox 755 0 0
slink /usr/bin/lpr /bin/busybox 755 0 0
slink /bin/ls /bin/busybox 755 0 0
slink /bin/lsattr /bin/busybox 755 0 0
slink /sbin/lsmod /bin/busybox 755 0 0
slink /usr/bin/lzmacat /bin/busybox 755 0 0
slink /bin/lzop /bin/busybox 755 0 0
slink /usr/bin/lzopcat /bin/busybox 755 0 0
slink /sbin/makedevs /bin/busybox 755 0 0
slink /bin/makemime /bin/busybox 755 0 0
slink /sbin/man /bin/busybox 755 0 0
slink /usr/bin/md5sum /bin/busybox 755 0 0
slink /sbin/mdev /bin/busybox 755 0 0
slink /usr/bin/mesg /bin/busybox 755 0 0
slink /usr/bin/microcom /bin/busybox 755 0 0
slink /bin/mkdir /bin/busybox 755 0 0
slink /sbin/mkdosfs /bin/busybox 755 0 0
slink /usr/bin/mkfifo /bin/busybox 755 0 0
slink /sbin/mkfs.minix /bin/busybox 755 0 0
slink /sbin/mkfs.vfat /bin/busybox 755 0 0
slink /bin/mknod /bin/busybox 755 0 0
slink /usr/bin/mkpasswd /bin/busybox 755 0 0
slink /sbin/mkswap /bin/busybox 755 0 0
slink /bin/mktemp /bin/busybox 755 0 0
slink /sbin/modprobe /bin/busybox 755 0 0
slink /bin/more /bin/busybox 755 0 0
slink /bin/mount /bin/busybox 755 0 0
slink /bin/mountpoint /bin/busybox 755 0 0
slink /bin/mt /bin/busybox 755 0 0
slink /bin/mv /bin/busybox 755 0 0
slink /sbin/nameif /bin/busybox 755 0 0
slink /usr/bin/nc /bin/busybox 755 0 0
slink /bin/netstat /bin/busybox 755 0 0
slink /bin/nice /bin/busybox 755 0 0
slink /usr/bin/nmeter /bin/busybox 755 0 0
slink /usr/bin/nohup /bin/busybox 755 0 0
slink /usr/bin/nslookup /bin/busybox 755 0 0
slink /usr/bin/od /bin/busybox 755 0 0
slink /usr/bin/openvt /bin/busybox 755 0 0
slink /usr/bin/passwd /bin/busybox 755 0 0
slink /usr/bin/patch /bin/busybox 755 0 0
slink /usr/bin/pgrep /bin/busybox 755 0 0
slink /bin/pidof /bin/busybox 755 0 0
slink /bin/ping /bin/busybox 755 0 0
slink /bin/ping6 /bin/busybox 755 0 0
slink /bin/pipe_progress /bin/busybox 755 0 0
slink /sbin/pivot_root /bin/busybox 755 0 0
slink /usr/bin/pkill /bin/busybox 755 0 0
slink /usr/sbin/popmaildir /bin/busybox 755 0 0
slink /sbin/poweroff /bin/busybox 755 0 0
slink /bin/printenv /bin/busybox 755 0 0
slink /usr/bin/printf /bin/busybox 755 0 0
slink /bin/ps /bin/busybox 755 0 0
```

```
slink /usr/bin/pscan /bin/busybox 755 0 0
slink /bin/pwd /bin/busybox 755 0 0
slink /sbin/raidautorun /bin/busybox 755 0 0
slink /usr/sbin/rdate /bin/busybox 755 0 0
slink /usr/sbin/rdev /bin/busybox 755 0 0
slink /usr/bin/readahead /bin/busybox 755 0 0
slink /usr/bin/readlink /bin/busybox 755 0 0
slink /usr/sbin/readprofile /bin/busybox 755 0 0
slink /usr/bin/realpath /bin/busybox 755 0 0
slink /sbin/reboot /bin/busybox 755 0 0
slink /bin/reformime /bin/busybox 755 0 0
slink /usr/bin/renice /bin/busybox 755 0 0
slink /usr/bin/reset /bin/busybox 755 0 0
slink /usr/bin/resize /bin/busybox 755 0 0
slink /bin/rm /bin/busybox 755 0 0
slink /bin/rmdir /bin/busybox 755 0 0
slink /sbin/rmmod /bin/busybox 755 0 0
slink /sbin/route /bin/busybox 755 0 0
slink /usr/bin/rtcwake /bin/busybox 755 0 0
slink /bin/run-parts /bin/busybox 755 0 0
slink /sbin/runlevel /bin/busybox 755 0 0
slink /usr/bin/runsv /bin/busybox 755 0 0
slink /usr/bin/runsvdir /bin/busybox 755 0 0
slink /usr/bin/rx /bin/busybox 755 0 0
slink /usr/bin/script /bin/busybox 755 0 0
slink /bin/scriptreplay /bin/busybox 755 0 0
slink /bin/sed /bin/busybox 755 0 0
slink /usr/sbin/sendmail /bin/busybox 755 0 0
slink /usr/bin/seq /bin/busybox 755 0 0
slink /bin/setarch /bin/busybox 755 0 0
slink /sbin/setconsole /bin/busybox 755 0 0
slink /usr/sbin/setfont /bin/busybox 755 0 0
slink /usr/bin/setkeycodes /bin/busybox 755 0 0
slink /usr/sbin/setlogcons /bin/busybox 755 0 0
slink /usr/bin/setsid /bin/busybox 755 0 0
slink /usr/bin/setuidgid /bin/busybox 755 0 0
slink /bin/sh /bin/busybox 755 0 0
slink /usr/bin/sha1sum /bin/busybox 755 0 0
slink /usr/bin/sha256sum /bin/busybox 755 0 0
slink /usr/bin/sha512sum /bin/busybox 755 0 0
slink /usr/bin/showkey /bin/busybox 755 0 0
slink /sbin/slattach /bin/busybox 755 0 0
slink /bin/sleep /bin/busybox 755 0 0
slink /usr/bin/softlimit /bin/busybox 755 0 0
slink /usr/bin/sort /bin/busybox 755 0 0
slink /usr/bin/split /bin/busybox 755 0 0
slink /sbin/start-stop-daemon /bin/busybox 755 0 0
slink /bin/stat /bin/busybox 755 0 0
slink /usr/bin/strings /bin/busybox 755 0 0
slink /bin/stty /bin/busybox 755 0 0
slink /bin/su /bin/busybox 755 0 0
slink /sbin/sulogin /bin/busybox 755 0 0
slink /usr/bin/sum /bin/busybox 755 0 0
slink /usr/bin/sv /bin/busybox 755 0 0
slink /usr/sbin/svlogd /bin/busybox 755 0 0
slink /sbin/swapoff /bin/busybox 755 0 0
slink /sbin/swapon /bin/busybox 755 0 0
slink /sbin/switch_root /bin/busybox 755 0 0
slink /bin/sync /bin/busybox 755 0 0
slink /sbin/sysctl /bin/busybox 755 0 0
slink /sbin/syslogd /bin/busybox 755 0 0
slink /usr/bin/tac /bin/busybox 755 0 0
slink /usr/bin/tail /bin/busybox 755 0 0
slink /bin/tar /bin/busybox 755 0 0
slink /usr/bin/tcpsvd /bin/busybox 755 0 0
slink /usr/bin/tee /bin/busybox 755 0 0
slink /usr/bin/telnet /bin/busybox 755 0 0
```

```
534 slink /usr/sbin/telnetd /bin/busybox 755 0 0
535 slink /usr/bin/test /bin/busybox 755 0 0
536 slink /usr/bin/tftp /bin/busybox 755 0 0
537 slink /usr/bin/tftpd /bin/busybox 755 0 0
538 slink /usr/bin/time /bin/busybox 755 0 0
539 slink /usr/bin/timeout /bin/busybox 755 0 0
540 slink /usr/bin/top /bin/busybox 755 0 0
541 slink /bin/touch /bin/busybox 755 0 0
542 slink /usr/bin/tr /bin/busybox 755 0 0
543 slink /usr/bin/traceroute /bin/busybox 755 0 0
544 slink /bin/true /bin/busybox 755 0 0
545 slink /usr/bin/tty /bin/busybox 755 0 0
546 slink /usr/bin/ttysize /bin/busybox 755 0 0
547 slink /sbin/tunctl /bin/busybox 755 0 0
548 slink /sbin/udhcpc /bin/busybox 755 0 0
549 slink /usr/sbin/udhcpd /bin/busybox 755 0 0
550 slink /usr/bin/udpsvd /bin/busybox 755 0 0
551 slink /bin/umount /bin/busybox 755 0 0
552 slink /bin/uname /bin/busybox 755 0 0
553 slink /bin/uncompress /bin/busybox 755 0 0
554 slink /usr/bin/unexpand /bin/busybox 755 0 0
555 slink /usr/bin/uniq /bin/busybox 755 0 0
556 slink /usr/bin/unix2dos /bin/busybox 755 0 0
557 slink /usr/bin/unlzma /bin/busybox 755 0 0
558 slink /usr/bin/unlzop /bin/busybox 755 0 0
559 slink /usr/bin/unzip /bin/busybox 755 0 0
560 slink /usr/bin/uptime /bin/busybox 755 0 0
561 slink /bin/usleep /bin/busybox 755 0 0
562 slink /usr/bin/uudecode /bin/busybox 755 0 0
563 slink /usr/bin/uuencode /bin/busybox 755 0 0
564 slink /sbin/vconfig /bin/busybox 755 0 0
565 slink /bin/vi /bin/busybox 755 0 0
566 slink /usr/bin/vlock /bin/busybox 755 0 0
567 slink /usr/bin/volname /bin/busybox 755 0 0
568 slink /bin/watch /bin/busybox 755 0 0
569 slink /sbin/watchdog /bin/busybox 755 0 0
570 slink /usr/bin/wc /bin/busybox 755 0 0
571 slink /usr/bin/wget /bin/busybox 755 0 0
572 slink /usr/bin/which /bin/busybox 755 0 0
573 slink /usr/bin/who /bin/busybox 755 0 0
574 slink /usr/bin/whoami /bin/busybox 755 0 0
575 slink /usr/bin/xargs /bin/busybox 755 0 0
576 slink /usr/bin/yes /bin/busybox 755 0 0
577 slink /bin/zcat /bin/busybox 755 0 0
578 slink /sbin/zcip /bin/busybox 755 0 0
579
580 # eof
```

Mit der im Ordner */initramfs* liegenden Datei *init.sh* wird das Dateisystem gemountet, werden Kernelmodule geladen und eine Shell angestartet.
Hier können auch Applikationen, wie z.B. unser Framework angestartet werden.

```
1  #!/bin/sh
2
3  # mount pseude filesystems
4  /bin/mount -t proc none /proc
5  /bin/mount -t sysfs none /sys
6  /bin/mount -t tmpfs tmpfs /dev/shm
7  /bin/mount -t tmpfs tmpfs /tmp
8
9  ## input modules
10 /sbin/insmod /modules/nls_base.ko
11 /sbin/insmod /modules/hid.ko
12 /sbin/insmod /modules/psmouse.ko
13 /sbin/insmod /modules/atkbd.ko
```

```
/sbin/insmod /modules/mousedev.ko

## serial device modules
/sbin/insmod /modules/serial_core.ko
/sbin/insmod /modules/8250.ko
/sbin/insmod /modules/8250_pci.ko

## usb modules including gps support
/sbin/insmod /modules/usbcore.ko
/sbin/insmod /modules/usbhid.ko
/sbin/insmod /modules/ehci-hcd.ko
/sbin/insmod /modules/ohci-hcd.ko
/sbin/insmod /modules/uhci-hcd.ko
/sbin/insmod /modules/cdc-acm.ko

## persitency modules
/sbin/insmod /modules/scsi_mod.ko
/sbin/insmod /modules/scsi_wait_scan.ko
/sbin/insmod /modules/sd_mod.ko
/sbin/insmod /modules/libata.ko
/sbin/insmod /modules/ahci.ko
/sbin/insmod /modules/ext2.ko

## some more stuff
/sbin/insmod /modules/i2c-core.ko
/sbin/insmod /modules/i2c-algo-bit.ko
/sbin/insmod /modules/jbd2.ko
/sbin/insmod /modules/drm.ko
/sbin/insmod /modules/crc16.ko
/sbin/insmod /modules/mbcache.ko
/sbin/insmod /modules/ext4.ko



echo ------------------------------------------
echo Welcome
echo ------------------------------------------

export PATH=/bin:/sbin

exec /bin/sh
```

Die Datei *buildxserver.sh* installiert nach dem Starten alle für das Compilieren der XServers benötigten Anwendungen. Hierbei ist darauf zu achten, dass es von Ditribution zu Distribution verschieden ist, welche Anwendungen benötigt werden. Unsere Konfiguration ist nur für Ubuntu Karmic Koala geeignet. Möchte man den XServer auf einer anderen Distribution compilieren, dann muss unter http://wiki.x.org/wiki/RequiredPackages nachgesehen werden, welche Applikationen man braucht. Sind die Programme installiert, wird mit git://anongit.freedesktop ein Git Repository auf den Rechner kopiert. Dieses enthält wiederum ein Buildscript, welches alle noch benötigten Sourcen herunterläd und compiliert.

```
echo '############################################################'
echo '###  Minimalanforderungen für Ubuntu Karmic Koala  ########'
echo '###  für andere Distributionen hier nachsehen:     ########'
echo '###  http://wiki.x.org/wiki/RequiredPackages       ########'
echo '############################################################'

##CPUCORES=4
CPUCORES=`cat /proc/cpuinfo | grep '^processor' | wc -l`
BUILDTASK=`echo ${CPUCORES}*2 | bc`
BUILDLOG=`pwd`/buildxserver.log
SOURCEDIR=`pwd`/source/xserver
TARGETDIR=`pwd`/image/xserver
##TARGETDIR=$HOME/xserver  ## legacy ;-)
```

```
14 REQUIRED="asciidoc autoconf automake autotools-dev bison docbook-utils doxygen flex fontconfig gcc
       gettext gettext-base git-core gperf groff intltool jadetex libfontconfig1-dev libfreetype6
       libfreetype6-dev libglib2.0-dev libncurses5-dev libpng12-0 libpng12-dev libssl-dev libtool
       linuxdoc-tools linuxdoc-tools-text linuxdoc-tools-latex m4 openssl perl pkg-config xmlto xmltex
       zlib1g zlib1g-dev"
15
16 echo ">>> prepare system (install required packages)"
17 sudo apt-get install $REQUIRED
18
19 echo '############################################################'
20 echo '###  BuildScript von Git Repository ausführen      ########'
21 echo '############################################################'
22
23 echo ">>> build xserver using target directory [$TARGETDIR}"
24 mkdir -p $TARGETDIR
25
26 echo ">>> fetch build script"
27 mkdir -p $SOURCEDIR
28 cd $SOURCEDIR
29 git clone git://anongit.freedesktop.org/git/xorg/util/modular util/modular
30
31 echo ">>> now build the xserver optimized for $CPUCORES cores (see log [$BUILDLOG])"
32 echo "#### starting build on `date` " > $BUILDLOG
33 echo "#### machine is `uname -a`" >> $BUILDLOG
34 echo >> $BUILDLOG
35 MAKEFLAGS="-j$BUILDTASK" ./util/modular/build.sh -n --clone $TARGETDIR &>> $BUILDLOG
36
37 echo ">>> done."
```

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[BBB+09] Bartel, Christian, Andreas Bannach, Tobias Braun, Andreas Brust, Antje Gloystein, Markus Göttle, Jens Lorek, Christian Müller, Marco Münch, Marcus von Rohden, Michael Roth und Julia Runge: *Praktikumsbericht Navi3D*. PDF, 2009. Perforce im Ordner SS2009.

[Bus10] Busybox, Erik Andersen: *BusyBox, Version 1.15.2*. Website, 2010. Online erreichbar unter `http://busybox.net/downloads/busybox-1.15.2.tar.bz2`; letzter Besuch am 18.02.2010.

[Fre10] Freedesktop.org: *Freedesktop, Buildsh*. Website, 2010. Online erreichbar unter `git://anongit.freedesktop.org/git/xorg/util/modular/build.sh`; letzter Besuch am 18.02.2010.

[Ker10] Kernel.org: *Linux Kernel, Version 2.6.32*. Website, 2010. Online erreichbar unter `http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.32.tar.bz2`; letzter Besuch am 18.02.2010.

[QNX10] QNX: *QNX Neutrino RTOS Overview*. Website, 2010. Online erreichbar unter `http://www.qnx.com/products/neutrino_rtos/`; letzter Besuch am 10.02.2010.

[Ram10] Ramesh, Balaji: *Error: CanŽt clobber writable file Perforce*. Website, 2010. Online erreichbar unter `http://balajiramesh.wordpress.com/2008/03/07/error-cant-clobber-writable-file-perforce/`; letzter Besuch am 15.02.2010.

[Tho10] Thomason, Lee: *TINYXML*. Website, 2010. Online erreichbar unter `http://www.grinninglizard.com/tinyxml/`; letzter Besuch am 26.02.2010.

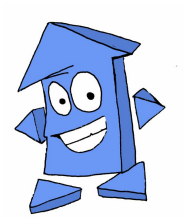
[uDMW10] Dr. Michael Wössner, Dr. Anja Köhne und: *NMEA-0183 Daten*. Website, 2010. Online erreichbar unter `http://www.kowoma.de/gps/zusatzerklaerungen/NMEA.htm`; letzter Besuch am 18.02.2010.

[Wik10a] Wiki, DGL: *glOrtho*. Website, 2010. Online erreichbar unter `http://wiki.delphigl.com/index.php/glOrtho/`; letzter Besuch am 16.02.2010.

[Wik10b] Wikipedia: *Perforce Jam*. Website, 2010. Online erreichbar unter `http://de.wikipedia.org/wiki/Perforce_Jam/`; letzter Besuch am 10.02.2010.

Autoren

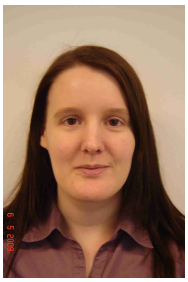
Christian Bartel, B.Sc.



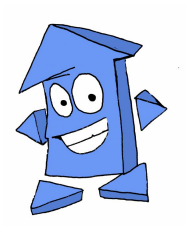
Achim Brauer, B.Sc.

Tobias Braun, B.Sc.

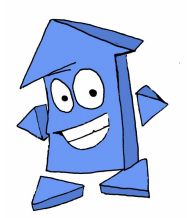
Andreas Brust, B.Sc.



Antje Gloystein, B.Sc.



Sebastian Hohmann, B.Sc.

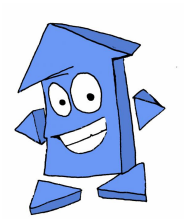


Thi Dieu Hien Le, B.Sc.

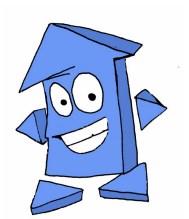
Christian Müller,B.Sc.

Dipl.-Inf.(FH)
Marcus von Rohden

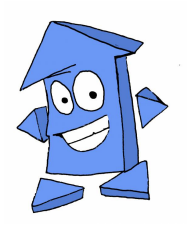
Julia Runge, B.Sc.



Johannes Seidel, B.Sc.



Alexander Steiger, B.Sc.



Andreas Völlger, B.Sc.