Ingenieurfakultät Bau Geo Umwelt
Lehrstuhl für Computergestützte Modellierung und Simulation
Prof. Dr.-Ing. André Borrmann

Intuitiver Lärmschutzwandentwurf entlang einer Trasse in frühen Entwurfsphasen

Fabian Pfitzner

Bachelorthesis
für den Bachelor of Science Studiengang Bauingenieurwesen

Autor:	Fabian Pfitzner
Matrikelnummer:	[redacted]
Betreuer:	Prof. Dr.-Ing. André Borrmann
	Štefan Jaud
Ausgabedatum:	12. November 2018
Abgabedatum:	12. April 2019

Abstract

The increasing growth of infrastructure in both urban and rural areas is causing some problems. One of them is noise. With the help of structural measures, one wants to oppose this. However, this often turns out to be more difficult and more expensive than expected. A decisive factor for the correct structural implementation of a sound insulation wall is the planning. This often results in as difficult due to very complex calculations. Thus, in earlier work phases, it is difficult to estimate whether an sound insulation measure is required or not. This makes proper routing more difficult. With the help of my work, these calculations can be more easily thought through and presented innovatively. With this tool it is possible to carry out fast calculations in order to to get a better overview of the noise propagation along the alignment. Noise barriers are simulated by means of blocks, whereby an image can be created for the shielding. It offers both an extension of the InteractiveAlignment, as well as the possibility to continue using its functions. When viewed completely, it is possible to carry out a noise calculation and display it in real time along a newly created traffic route.

Zusammenfassung

Der zunehmende Wachstum der Infrastruktur in sowohl städtischen, als auch ländlichen Gebieten zieht einige Probleme mit sich. Eines davon ist Lärm. Mithilfe von baulichen Maßnahmen will man diesem entgegentreten. Doch oftmals gestaltet sich das als schwieriger und aufwendiger als gedacht. Ein entscheidender Faktor für die richtige bauliche Umsetzung einer Schallschutzwand ist die Planung. Diese ergibt sich häufig als langwierig durch sehr komplexe Berechnungen. Somit ist es in früheren Leistungsphasen mühsam abzuschätzen, ob eine Schallschutzmaßnahme benötigt wird oder nicht. Dadurch erschwert sich eine ordnungsgemäße Trassierung. Mithilfe meiner Arbeit sollen diese Berechnungen einfacher durchdacht und innovativ dargestellt werden. Mit dem Lärmtool ist es möglich schnelle Berechnungen durchzuführen, um einen besseren Überblick über die Lärmausbreitung an der Trasse zu bekommen. Anhand von Blöcken werden Lärmschutzwände simuliert, wodurch ein Bild für die Abschirmung geschaffen werden kann. Es bietet sowohl eine Erweiterung des Interactive Alignment, als auch eine Möglichkeit dessen Funktionen weiter zu verwenden. Vollständig betrachtet ergibt sich die Möglichkeit, eine Lärmberechnung durchzuführen und diese in Echtzeit entlang eines neu erstellten Verkehrsweges darzustellen.

Inhaltsverzeichnis

Abkürzungsverzeichnis

2D	Zweidimensional
3D	Dreidimensional
BIM	Building Information Modeling
IFC	Industry Foundation Classes
DTV	durchschnittlicher täglicher Verkehr
KFZ	Kraftfahrzeug
GIS	Geographic Information Systems
CDP	Collaborative Design Plattform
dB	Dezibel
PFV	Planfeststellungsverfahren
UVP	Umweltverträglichkeitsprüfung
BREP	Boundary Representation
DTM	Digitales Gelände Modell
TIN	Triangulated Irregular Network
FNP	Flächennutzungsplan
AEC	Arichtecture, Engineering and Construction
VR	Virtual Reality

Kapitel 1

Einführung

1.1 Problemstellung

Ohne den Ausbau von Lärmschutzwänden wäre eine Erweiterung des infrastrukturellen Sektors kaum möglich. Deswegen werden seit 35 Jahren in Deutschland Lärmschutzwände gebaut. Staatsbauämter haben frühzeitig erkannt, dass bauliche Anlagen angrenzende Umfelder entscheidend beeinflussen können und sehen eine gute Gestaltung dieser als Aufwertung des Verkehrswegs. Um die Bürger vor Lärm zu schützen sind verbindlichen Grenzwerte auf Grundlage des Bundesimmissionsschutzgesetzes vorgeschrieben (SSF-Ingenieure, 2018).

Abbildung 1.1: Lärmschutzwand München-Freimann (Ingenieurbüro Grassl, 2018)

Für eine Einschätzung über die Einhaltung der Werte müssen verschiedene Parameter des Schalls berechnet werden. Herkömmliche Methoden der Lärmberechnung gestalten sich oft als umständlich und langwierig. Das liegt zum einem an der Vielzahl der Einflussparameter und zum anderen an den zu hoch gestellten Anforderungen. Gerade in früheren Leistungsphasen ist es oftmals nicht erforderlich jedes kleinste Detail in die Berechnungen miteinfließen zu lassen, da dies die Zielsetzung häufig nicht verlangt. Es ist vielmehr wichtig einen schnellen Überblick zu schaffen, um zu erkennen, ob eine Baumaßnahme erforderlich ist oder nicht. Damit eine schnelle Lärmberechnung an der Trasse durchgeführt werden kann, bedarf es

nur an wenigen Parameter, welche sich auch noch in späteren Leistungsphasen erweitern lassen. Um unnötige Genauigkeiten zu vermeiden ist eine Großzahl der Kenngrößen, wie beispielsweise die Oberflächenbeschaffenheit der Straße, für die Berechnungen zunächst nicht relevant. Dieser Gedanke wird sich durch den Verlauf der Arbeit ziehen und ist mit folgender Zielsetzung gekoppelt: Das Tool soll keine möglichst exakten Ergebnisse erzielen, sondern kreativ und innovativ einen schnellen, variablen Überblick über die Lärmsituation verschaffen.

1.2 Motivation

Die Digitalisierung in der Baubranche spielt eine immer größerwerdende Rolle. Einzelne Prozesse können mithilfe von Software unterstützt werden. Dadurch kann eine Optimierung von den unterschiedlichsten Vorgehensweisen erreicht werden. Im Bereich der Lärmermittlung gibt es viele Angriffspunkte für derartige Optimierungskonzepte. Die Motivation bei der Erstellung dieser Arbeit und bei der Entwicklung dieses Plugins ergab sich aus den oben beschriebenen Problemen und zukünftigen Möglichkeiten. Eine einfache und schnelle Lärmberechnung durchzuführen ist eine große Erleichterung von vielen Teilprozessen und könnte zukünftig die Planung und den Entwurf einer baulichen Schallschutzeinrichtung grundsätzlich verändern. Des Weiteren könnte damit im ganzheitlichen der Prozess Building Information Modeling (BIM), welcher die Zukunft der Baubranche beschreibt, weiter voran getrieben werden. Im folgenden Kapitel wird dieser Prozess genauer definiert und beschrieben.

1.3 Ziel der Arbeit

Bei der Planung eines Verkehrsweges spielen verschiedene räumliche Faktoren eine Rolle, wie z.B. Biotope, Siedlungen und Wälder. Im Rahmen der wissenschaftlichen Arbeit über den Achsentwurf und des Interactive Alignment Design Tools von Schlenger (2018) wurde sich mit solchen Faktoren und der Anpassung der Trasse an ein Gelände beschäftigt. Hier ist jedoch zu berücksichtigen, dass auch weitere Einflussparameter überprüft werden müssen.

Dabei spielt vor allem die durch den Verkehr entstehende Lärmentwicklung eine große Rolle. Um die zu erwartende Lärmbelastung, für die eventuelle Einleitung entsprechender Gegenmaßnahmen abzuschätzen, ist eine Lärmsimulation nötig. Aufbauend auf den Resultaten von Jonas Schlenger soll eine solche Simulation an der Trasse entstehen. In dieser Arbeit sollen Untersuchungen zu folgenden Fragestellungen durchgeführt werden:

- Welche Parameter und Methoden existieren für schnelle Lärmberechnungen?
- Wie lässt sich die Lärmentwicklung entlang eines Verkehrsweges grafisch darstellen?

- Welche Möglichkeiten bieten sich um eine pauschale Lärmberechnung in der frühen Entwurfsphase durchzuführen?

- Welchen Einfluss haben Hindernisse auf die Ausbreitung der Schallwellen?

Des Weiteren soll in dieser Arbeit ein Prototyp einer Lärmsimulation an einer Trasse auf Basis der Ergebnisse des Interactive Alignment Tools entwickelt werden.

Kapitel 2

Grundlagen

2.1 Entwurf

2.1.1 CDP-Table

Die Collaborative Design Plattform (CDP) bietet die Möglichkeit Echtzeit Computer Analysen intuitiv durchzuführen. Der Tisch basiert auf zwei Prinzipien. Zum einen fördert er die Einbettung des kreativen Prozesses. Somit können Simulationen und Analysen bereits in den frühen Phasen miteinbezogen werden. Des Weiteren bietet er eine einfache Erstellung und Gestaltung für Tools. Ein Beispiel dafür ist die Applikation der visuellen Programmiersprache, was es dem Nutzer auch ohne Code möglich macht, Analysen und Simulationen zu implementieren. Der Grundgedanke des Tisches ist es, eine interaktive Plattform für frühe Leistungsphasen zu bieten. Dabei liegt der Fokus darauf physische Modelle, wie beipielsweise Blöcke, mit einer Touchoberfläche zu koppeln. Veränderungen des Modell haben dabei direkten Einfluss auf die Simulation in Echtzeit. Die berechneten Ergebnisse können ebenfalls in realer Zeit angezeigt werden (Schubert & Petzold, 2017).

Die Hardware wurde unter Berücksichtigung der oben genannten Prinzipien entwickelt. Das System ist ein Eigenbau und bietet als Basis eine Multitouch-Oberfläche. Mithilfe der Diffused Illumination (DI) Technik wird das Erkennen jeglicher Objekte ermöglicht. Der interaktive Tisch besitzt eine matte Projektionsfläche (A), auf die ein Bild von unten projiziert wird. Das Projektionsbild (B) wird über einen Spiegel (C) umgeleitet. Mit Infrarot-Strahlern (D) wird die Projektionsfläche ausgeleuchtet. Die Unterseite der Projektionsfläche (A) wird über den Spiegel (C) mit einer Infrarotkamera (E) aufgenommen. Auf dem IR-Kamerabild sind Gegenstände und Berührungen sichtbar. Mithilfe der Microsoft Kinect Kamera (I) können auf dem Tisch platzierte Objekte als 3D-Punktwolke eingescannt werden. In Verbindung mit dem Kamerabild (E) ist es so möglich ein 3D-Modell zu erstellen. Ein Rechner (F)

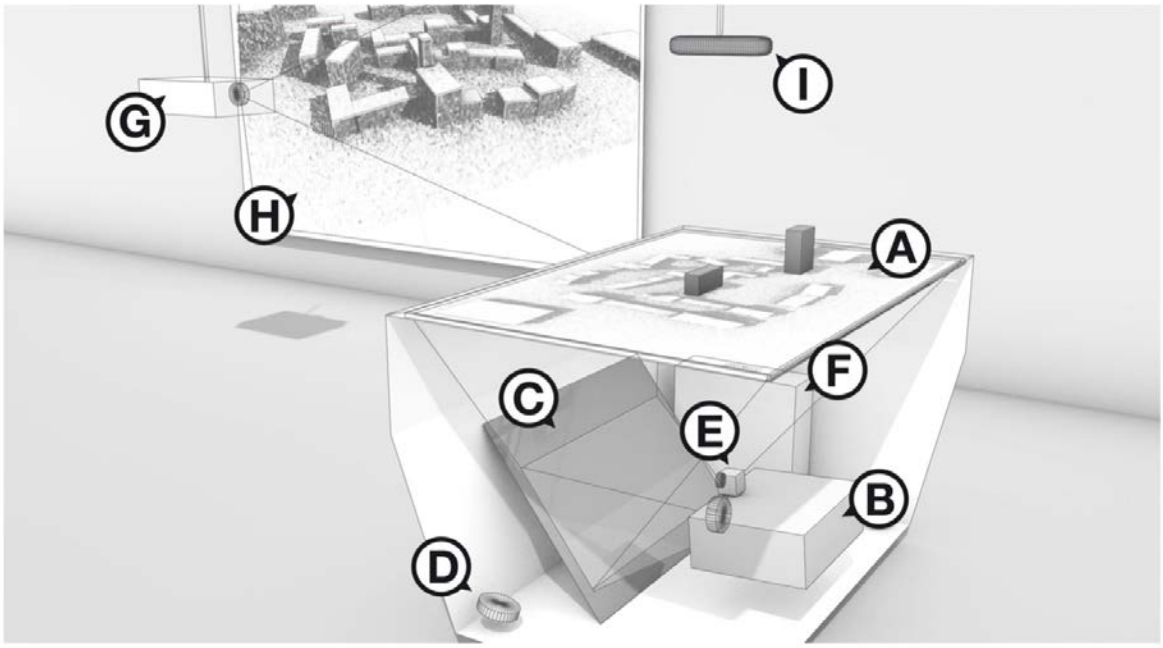


Abbildung 2.1: CDP-Tisch mit Projektionsfläche (A), Projektionsbild (B), Spiegel (C), Infrarot-Strahlern (D), Infrarotkamera (E), Rechner (F), Microsoft Kinect Kamera (I) (Schubert, 2012)

bearbeitet die Kameradaten für das Projektionsbild (B). Mittels einer weiteren Grafikkarte können weitere Ausgaben abgebildet werden(Projektor(G), Leinwand(H)). Neben der Anzeige von zweidimensionalen Informationen besteht ebenfalls die Möglichkeit eine dreidimensionale Darstellung des Entwurfs zum besseren Verständnis abzubilden (Schubert & Petzold, 2017).

Die Anbindung der Plugins erfolgt über eine Middleware (geschrieben in C++). Durch das Plug-in Framework verlaufen Datenkommunikation, Interaktion, als auch die Visualisierung der Berechnungsergebnisse komplett über die Host-Anwendung, welche die erforderlichen Daten zu Verfügung stellt (Schubert, 2012).

2.1.2 Interactive Alignment Plugin

Das Interactive Alignment Plugin konzentriert sich darauf, einen Achsentwurf innovativ in frühen Leistungsphasen zu erstellen. In Verschmelzung mit dem CDP -Table und der grafischen Oberfläche wird dies dem Benutzer ermöglicht. Mit flexiblen Optionen lässt sich das Herzstück jedes Infrastrukturprojekts, die Achse, beliebig platzieren und trassieren. Die Kurve wird dabei in einen horizontalen und vertikalen Teil zerlegt. Der horizontale Abschnitt ist durch die Projektion der 3D-Achse in die x,y Ebene beschrieben. Der vertikale Teil hingegen beschreibt den Verlauf der Kurve hinsichtlich der y,z Ebene (Markič *et al.*, 2018).

Jedes Alignment kann aus drei verschiedenen Teilsegmenten bestehen. Es gibt Geraden-, Kurven- und Klothoidenelemente (Übergangsbögen). Vor allem in der Modellierung, welche das Tool ebenfalls unterstützt, ist das wichtig. Mithilfe der einzelnen Segmente kann die Trasse gezeichnet werden. Die Besonderheit des Tools liegt allerdings in der Geländeanpassung. Es bietet die Möglichkeit ein Digitales Gelände Modell (DTM) durch Triangulated Irregular Network (TIN)-Geometrie einzulesen. Unter dem Triangulated Irregular Network versteht man ein Netz an 3D-Punkten, welche durch unregelmäßig geformte Dreiecke verbunden sind. Die minimalen und maximalen y-Werte der Punkte aus dem DTM sind dabei für die Höhenberechnung entscheidend. Ähnlich wie bei der horizontalen Ansicht wird die eine Gradiente nach der Berechnung aus verschiedenen Segmenten gezeichnet. Die fertig erstellte Achse kann als Trasse in einer IFC-4x1 Datei gespeichert werden. Damit wird eine breite Möglichkeit der Wiederverwendung in sämtlicher Software geboten. Die grafische Oberfläche wurde bei der Entwicklung des Tools einfach gehalten, damit keine unnötige Ablenkung des Benutzers erzeugt wird. Die Funktionen beschränken sich somit auf Hinzufügen, Auswählen, Bewegen und Löschen einzelner Achssegmente. Als Ergebnis bildet sich eine 3D-Polylinie welche das Alignment sowohl in einer horizontalen, als auch in einer vertikalen Sicht in einer unkomplizierten Form beschreibt (Markič *et al.*, 2018).

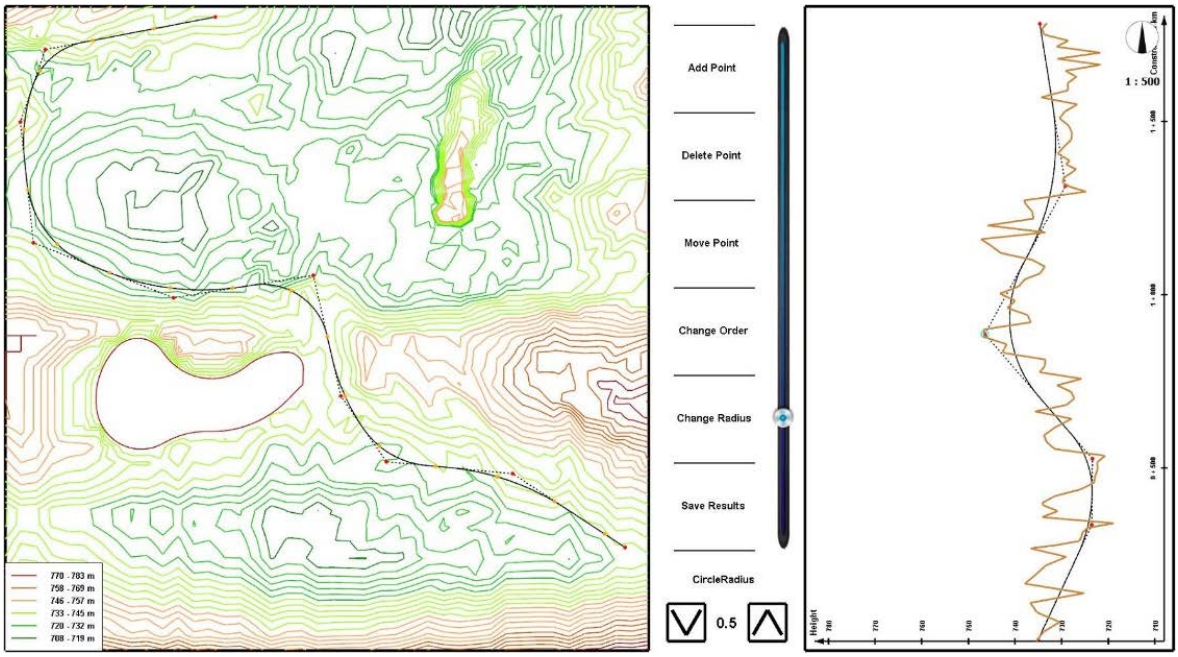


Abbildung 2.2: Interactive Alignment Plugin (Markič *et al.*, 2018)

Darauf aufbauend soll nun eine Erweiterung zur Durchführung von Lärmsimulationen entlang der Trasse entstehen. Fokus liegt dabei auch auf der Verwendung der einzelnen Funktionen des CDP-Tisches. Insgesamt soll ein fließender Übergang zwischen der Erstellung der Gradiente und Berechnung von Lärm entstehen.

2.2 Lärm

2.2.1 Allgemeine Begrifflichkeiten

Um ein besseres Verständnis über Lärm im Allgemeinen zu bekommen, ist zunächst die Definition einiger Begrifflichkeiten erforderlich. Grundsätzlich versteht man unter **Schall** elastodynamische Schwingungen und Wellen (Gemeinschaftsarbeitsausschuss NALS/NATG: Terminologie und Einheiten der Akustik, 2009).

Schalldruck ist von dem Mensch wahrgenommener Schall in einem Wertebereich von 0,0002 bis 20 Pascal. In der Psychophysik werden lineare Veränderungen von Reizempfindungen, wie dem Hören durch eine Vervielfachung der Reizstärke beschrieben. Der **Schalldruckpegel** ist eine solche logarithmische Adaption des Schalldrucks. **Isophonen** sind Kurven mit gleicher Lautstärke, wodurch sich unterschiedliche Schalldruckpegelbereiche bewerten lassen. International durchgesetzt hat sich die bewertete A-Kurve, welche aus der 40-phon Isophone abgeleitet wurde (Häupl *et al.*, 2017).

Der **Schallpegel** entspricht in der Richtlinie für Lärmschutz an Straßen (RLS-90) dem Schalldruckpegel und wird in Dezibel (dB) angegeben. Unter dem **A-Schallpegel** versteht man einen fequenzbewerteten Schallpegel, welcher durch die A-Bewertung die frequenzabhängige Empfindlichkeit des menschlichen Gehörs miteinbezieht. Dieser wird in dB(A) angegeben und ist nach DIN IEC 651 definiert (Bundesministerium für Verkehr, 1990).

Der **Mittelungpegel** gibt in einem zeitlich definierten Raum den Mittelwert des A-Schallpegels an. Dabei wird im Straßenverkehr die Unterscheidung zwischen Tag (16 Stunden, 6-22 Uhr) und Nacht (8 Stunden, 22-6 Uhr) getroffen. Der Mittelungspegel soll anhand einer Zahl die Belastung durch Störgeräusche charakterisieren. Dabei hat jedes Einzelgeräusch während der Beurteilungszeit einen Einfluss auf den Pegel. Der **Beurteilungspegel** entspricht bei Straßenverkehrsgeräuschen nahezu dem Mittelungspegel. Lediglich bei Kreuzungen unterscheidet dieser sich durch einen Zuschlag für erhöhte Störwirkungen vom Mittelungspegel (Bundesministerium für Verkehr, 1990).

Unter der **Schallemission** versteht man die Aussendung von Schall einer Schallquelle. Im Gegenzug ist die **Schallimmission** die Einwirkung von Schall auf ein bestimmtes Gebiet oder einen Punkt. Die Höhe der Immission ist dabei durch den Mittelungspegel definiert (Bundesministerium für Verkehr, 1990).

2.2.2 Umgebungslärm

> *„Umgebungslärm [sind] unerwünschte oder gesundheitsschädliche Geräusche im Freien, die durch Aktivitäten von Menschen verursacht werden, einschließlich des*

> *Lärms, der von Verkehrsmitteln, Straßenverkehr, Eisenbahnverkehr, Flugverkehr sowie Gelände für industrielle Tätigkeiten gemäß Anhang I der Richtlinie 96/61/EG des Rates vom 24. September 1996 über die integrierte Vermeidung“* (EU-Parlament, 2002, Artikel 3a)

Die EG-Umgebungslärmrichtlinie beschreibt einen Leitfaden zur Bewertung und Bekämpfung von diesem Umgebungslärm. Das Hauptziel liegt darin die schädlichen Auswirkungen von Umgebungslärm auf Betroffene zu verhindern. Für die Erreichung dieses Ziels wurden folgende Maßnahmen eingeleitet: (EU-Parlament, 2002)

- Erfassung von Lärmbelastungen in den Mitgliedsstaaten
- Informieren der Öffentlichkeit
- Aufstellen von Aktionsplänen bei problematischen Lärmsituationen

Zunächst einmal ist es wichtig die Lärmsituation zu erfassen. Dafür werden sowohl Messungen, als auch Berechnungen benötigt. Für ein aussagekräftiges Bild werden Informationen gesammelt und ausgewertet. Nach der Auswertung ist es notwendig die Öffentlichkeit fachlich aufzuklären und miteinzubeziehen. Bei problematischen Lärmsituationen sind Aktionspläne zu erstellen. Darin werden Abwicklungen zur Handhabung des Umgebungslärms und dessen Auswirkungen, einschließlich der Lärmminderung festgelegt. So soll eine Verbesserung der Geräuschsituation erzielt werden. Für die Umsetzung sind technische und planerische Maßnahmen erforderlich. Anschließend folgt eine Aufklärung der EU-Kommission über die ermittelten Ergebnisse (EU-Parlament, 2002).

Für einen schnellen Überblick, ob eine Maßnahme erforderlich ist oder nicht, bedarf es nicht an einer komplexen, tiefgreifenden Lärmberechnung. Anhand einfacher Strukturen soll die Möglichkeit geboten werden, Einflüsse von Lärmsanierungsmaßnahmen frühzeitig zu erkennen. Bisweilen gibt es nur begrenzte Möglichkeiten den Einfluss von Lärm in den frühen Leistungsphasen zu erkennen. Eine Möglichkeit davon sind die Lärmkarten des bayrischen Landesamt für Umwelt (Bayrisches Landesamt für Umwelt, 2018).

Lärmkarten

Ziel, gemäß der EU-Umgebungsrichtlinie, ist es die Informationen über die aktuelle Lärmsituation anhand eines Lärmindexes darzustellen. Dafür wurden die Lärmkarten vom Landesamt für Umwelt in Bayern ausgearbeitet. Diese beinhalten nicht nur die derzeitige Lärmsituation, welche mittels Isophonen dargestellt wird, sondern auch geltende Grenzwerte und die Anzahl der Personen in betroffenen Gebieten. Mit diesen Informationen können die einzelnen Zustände genauer analysiert werden (EU-Parlament, 2002).

Die Erstellung der Lärmkarten bildet sich für Ballungsräume mit über hunderttausend Einwohnern und Hauptverkehrsstraßen, mittels einer zweistufige Vorgehensweise. Diese wird seit 2012 im fünfjährigen Turnus überprüft (Bayrisches Landesamt für Umwelt, 2018).

Die Karten, inklusive der Werte sind in einer Datenbank gespeichert. Nach der Berechnung der Daten werden diese an ein sogenanntes Lärmbelastungskataster übermittelt. Mithilfe bestimmter Auswertesoftware lassen sich die dB-Werte analysieren und kategorisieren. Die Datensätze werden mit Zeitangaben gespeichert. Eine erneute Berechnung und Aktualisierung der Datenwerte ist beliebig oft wiederholbar, aber mit einem gewissem Aufwand verbunden. Die ermittelten Daten strecken sich über ein 10 m Raster und beinhalten alle Hauptverkehrsstraßen. Die dB-Werte werden darin für Tag und Nacht, ausgehend von 4 Meter über dem Gelände ermittelt. Bei der Berechnung wurden das Gelände, Bebauungen und Lärmschutzeinrichtungen miteinbezogen. Des Weiteren wurden die Parameter Geschwindigkeit und Fahrbahnoberfläche als schalltechnische Einflussgrößen berücksichtigt. In Großstädten wurden ebenfalls Straßenbahnnetze und oberirdisch verlaufende U-Bahnen miteinbezogen Die einzeln dargestellten Isophonenbänder sind dabei getrennt zu betrachten, mit Hinblick auf die Auswirkungen auf Menschenmassen, Schulen und Krankenhäusern (EU-Parlament, 2002).

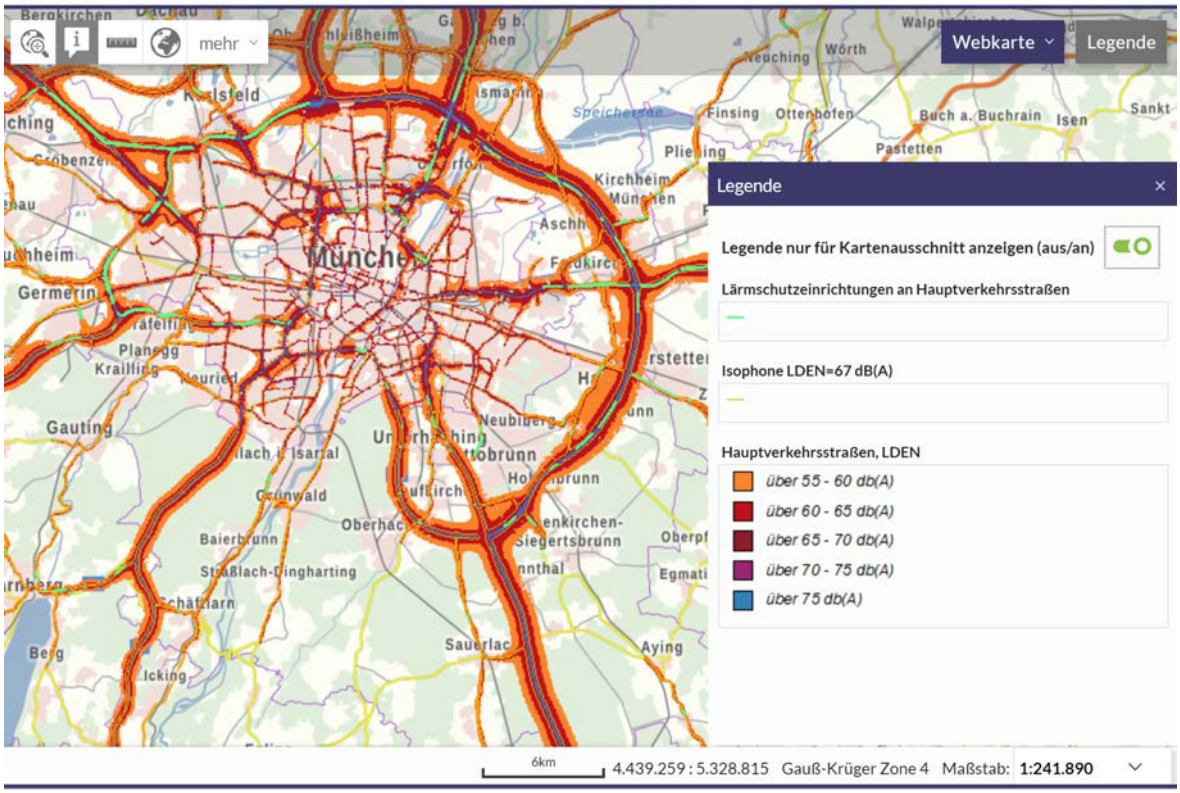


Abbildung 2.3: Lärmkarte für den Raum München (Bayrisches Landesamt für Umwelt, 2019)

Die Lärmkarten bieten einen gute Einschätzung über die Lärmsituation in verschiedenen Gebieten. Dennoch werden die Werte in den Karten nicht in Echtzeit berechnet. Von daher ist es nicht möglich, abzuschätzen was für Auswirkungen eine baulichen Schallschutzmaßnahme auf die Lärmumgebung hätte. Dort grenzt sich das im Rahmen dieser Arbeit entwickelte Lärmtool klar ab. Darin besteht die Möglichkeit eine Berechnung der aktuellen Lärmsituation in Echtzeit durchzuführen. Das hat zum Vorteil, dass weitere Einflüsse, wie beispielsweise eine bauliche Schallschutzmaßnahme, mit in die Bewertung einbezogen werden können.

2.2.3 Umsetzung von Lärmschutzmaßnahmen in der Praxis

Vor dem Neubau oder Umbau von Straßen, Schienenwegen und Lärmschutzsanierungen sind Planfeststellungsverfahren (PFV) durchzuführen. Dabei werden verschiedene gesetzliche Leitlinien beachtet. Das Verfahren endet mit einem Planfeststellungsbeschluss, der mit Auflagen versehen ist. Mit dem Planfeststellungsbeschluss endet ebenfalls die vierte Leistungsphase (Genehmigungsplanung). Die frühen Leistungsphasen sind somit entscheidend. Im Laufe des PFV sind die Schallansprüche und deren Umweltauswirkungen zu prüfen. Des Weiteren ist es nötig Betroffene rechtzeitig zu informieren. In der Gesamtheit des PFV sind Prüfungen für die Umweltverträglichkeit, Lärmvorsorge und Lärmsanierung zu erfüllen (Bayrisches Landesamt für Umwelt, 2018).

Umweltverträglichkeit

Eine Prüfung der Umweltverträglichkeit ist bei dem Bau von Bundesautobahnen und Bundesstraßen, wenn diese Schnellstraßen sind, zwangsweise erforderlich. Bei dem Bau von Schienenwegen ist eine Umweltverträglichkeitsprüfung (UVP) ebenfalls unumgänglich. Angesichts einer Veränderung bestehender Landesverkehrswege hängt die Pflicht einer Untersuchung von der Überschreitung bestimmter Größenwerte ab. Das Ziel der UVP ist es unmittelbare und mittelbare Auswirkungen eines Bauvorhabens zu bestimmen und diese zu beurteilen. Bei dem Bewertungsprozess wird die Öffentlichkeit miteinbezogen (EU-Parlament, 2002).

Teil der Prüfung ist ebenfalls die Betrachtung des Gesamtbeurteilungspegel. Dieser gibt in einem Beurteilungszeitraum den Mittelwert des Schallpegels an einem Verkehrsweg an. Damit werden die Auswirkungen auf die Umwelt ausreichend abgeschätzt. Bei einem Neubau oder einem erheblichem Umbau eines Verkehrsweges wird ein Vergleich der Mittelungspegel (entspricht bei Straßenverkehrsgeräuschen dem Beurteilungspegel) vor und nach der Fertigstellung durchgeführt (EU-Parlament, 2002).

Lärmvorsorgeansprüche

In PFV wird unterschieden zwischen einem Neubau und "wesentlichen Änderungen". Diese sind akustische Änderungen, die durch einen baulichen Eingriff entstehen können. Falls eine Erhöhung des Beurteilungspegel durch betriebliche Änderungen oder Ähnlichem verursacht wurde führt das nicht zu einem Schallschutzanspruch. Vorrangig ist der Bau von aktivem Schallschutz. Dieser bietet eine aktive Maßnahme gegen die Schallausbreitung. Beispiele dafür sind leise Straßenbeläge, Lärmschutzwälle und Schallschutzwände. Wenn dieser als unverhältnismäßig teuer eingestuft werden sollte, kann auch ein passiver Lärmschutz, wie beispielsweise ein Austausch von Fenstern in Betracht gezogen werden (EU-Parlament, 2002).

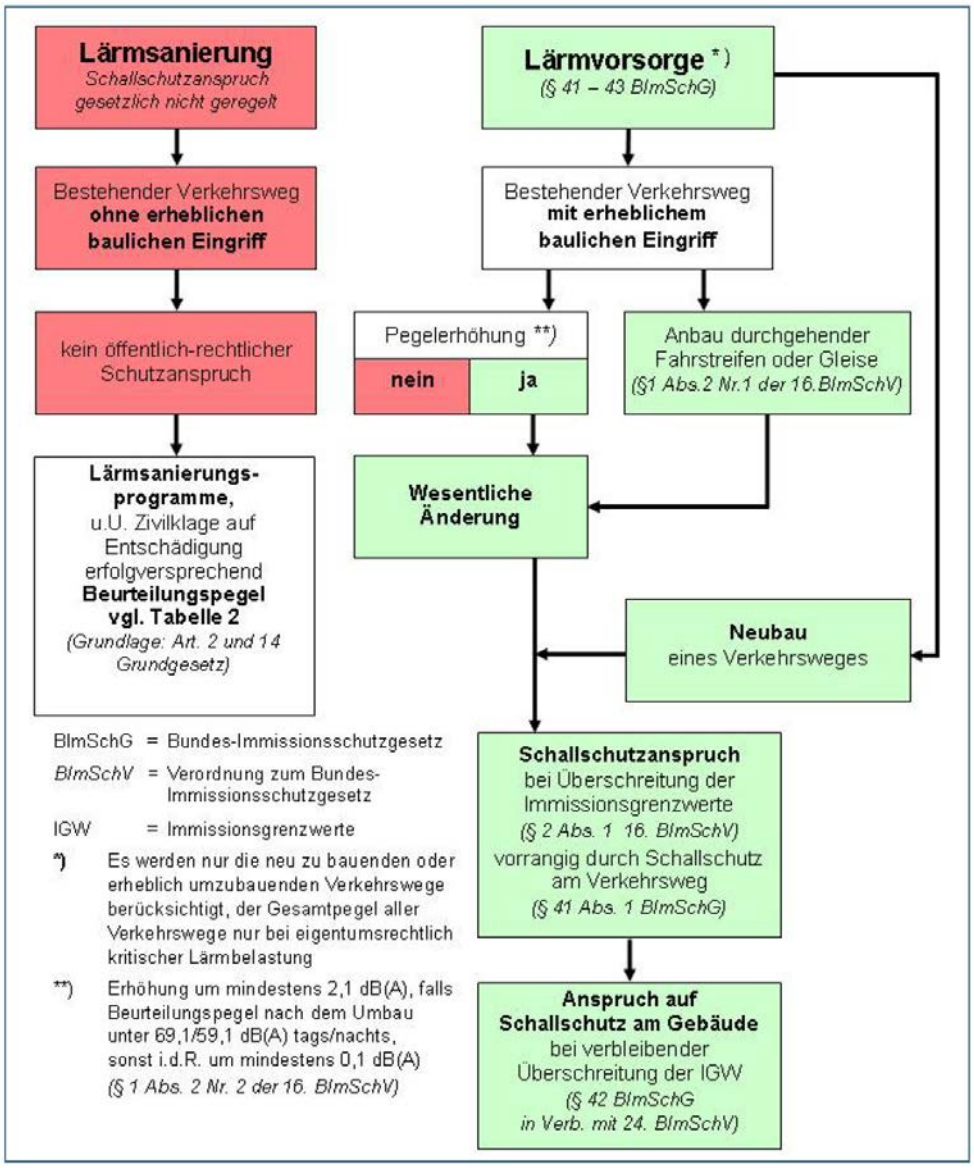


Abbildung 2.4: Lärmvorsorgeansprüche (Bayerisches Landesamt für Umwelt, 2017)

Die Berechnung der Grenzwerte erfolgt nach den in der Sechzehnten Verordnung zur Durchführung des Bundes-Immissionsschutzgesetzes definierten Regelwerken, welche unter anderem auch Verfahren der RLS-90 Norm beinhaltet. Dabei werden alle Immissionsorte im Bereich des Planfeststellungsbereiches eingeschlossen. Die Emissionsstellen werden meist wegführend an Streckenabschnitten, auch außerhalb des Bereichs miteinbezogen. Folglich ist für die Berechnung des Gesamtbeurteilungspegel an Immissionsorten immer der Verkehrsweg im Baubereich zu betrachten. Diese werden aber im Gegensatz zur Umweltverträglichkeitsprüfung nur unter oben genannten Bedingungen durchgeführt (Bayerisches Landesamt für Umwelt, 2017).

Folgende Grenzwerte sind dabei, entsprechend des Bayerisches Landesamt für Umwelt (2017) einzuhalten:

Art der Flächennutzung	**Immissionsgrenzwert (tags)**	**Immissionsgrenzwert (nachts)**
Krankenhäuser, Schulen, Kurheime und Altenheime	57 dB(A)	47 dB(A)
Reine und allgemeine Wohngebiete und Kleinsiedlungsgebiete	59 dB(A)	49 dB(A)
Kerngebiete, Dorfgebiete und Mischgebiete	64 dB(A)	54 dB(A)
Gewerbegebiete	69 dB(A)	59 dB(A)

Tabelle 2.1: Immissionsgrenzwerte an Verkehrswegen (Bayerisches Landesamt für Umwelt, 2017)

Lärmsanierung

Unter einer Lärmsanierung versteht man eine Erstellung einer Lärmschutzmaßnahme an bestehenden Verkehrswegen. Für Lärmsanierungen gibt es grundsätzlich keine verbindliche gesetzliche Regelungen in Deutschland. Nur unter Ausnahmefällen, wie eine Auswirkung der Baumaßnahmen auf Dritte, muss diese ins Planfeststellungsverfahren mit aufgenommen werden. Wie bei der Lärmvorsorge haben aktive Lärmschutzmaßnahmen ebenfalls Vorrang Die Auslösewerte für eine Lärmsanierung sind in folgender Tabelle definiert (Bayerisches Landesamt für Umwelt, 2017):

Art der Flächennutzung	**Beurteilungspegel (tags)**	**Beurteilungspegel (nachts)**
Krankenhäuser, Schulen, Kurheime und Altenheime, reine und allgemeine Wohngebiete sowie Kleinsiedlungsgebiete	67 dB(A)	57 dB(A)
Kerngebiete, Dorfgebiete und Mischgebiete	69 dB(A)	59 dB(A)
Gewerbegebiete	72 dB(A)	62 dB(A)

Tabelle 2.2: Beurteilungspegel für Lärmsanierung an Bundesfern- und Staatsstraßen (Bayerisches Landesamt für Umwelt, 2017)

2.2.4 Möglichkeiten zur Lärmminderung

Lärmschutztechnische Straßenplanung

Grundsätzlich ist es zunächst sinnvoll sich in den frühen Leistungsphasen mit der Lärmschutzplanung auseinanderzusetzen. Bei der Planung der Trasse sollte ein größtmöglicher Abstand zu den schutzbedürftigen Gebäuden gehalten werden. Damit können bauliche Maßnahmen bereits frühzeitig vermieden werden. Mit einer Abstandsverdoppelung kann eine Minderung um 4 dB erreicht werden (Bundesministerium für Verkehr, 1990).

Ebenfalls wichtig ist es die verschiedene Varianten einer Trasse auch hinsichtlich des Lärmschutzes abzuwägen. Durch anfahrende und abbremsende Fahrzeuge können starke Schallpegelerhöhungen entstehen. Deswegen sollte Knotenpunkte, wie Kreuzungen möglichst vermieden werden. Bei unausweichlichen Kreuzungen ist es lärmschutztechnisch ratsam die Hauptverkehrsstraße stets zu überführen, um so eine zusätzliche Schalldämmmaßnahme zu erhalten (Bundesministerium für Verkehr, 1990).

Bauliche Maßnahmen

Zunächst ist es wichtig die Straßenoberfläche zu betrachten. Neben ihrer herstellungsbedingten Toleranzen gibt es große Unterschiede hinsichtlich des verwendeten Materials und der Oberflächenbeschaffenheit. So ist beispielsweise zu beachten, dass sich die Straßenoberfläche im Laufe der Zeit verändert. Grundsätzlich sind Querrillen, Unebenheiten, Stufen, Schellen und regelmäßig profilierte Querschnitte in der Planung zu vermeiden. Eine wichtige Rolle spielt außerdem die standfestes Ausbildung des Straßenoberbaus (Bundesministerium für Verkehr, 1990).

Zusätzlich bieten sich neben der richtigen Gestaltung der Straßenoberfläche weitere bauliche Schutzmöglichkeiten. So gibt es Lärmschutzwälle, Lärmschutzwände, Einschnitts- und Troglagen und Teil- und Vollabdeckungen (Tunnel). Es ist ebenfalls möglich verschieden Varianten zu kombinieren. Besonders wichtig dabei ist es diese konstruktiven Maßnahmen möglichst unauffällig in das Orts- und Landschaftsbild einzufügen. Folglich ist ein Wall einer Lärmschutzwand in der Natur vorzuziehen. Des Weiteren müssen auch die akustischen Eigenschaften, die Standsicherheit, die bautechnische Ausbildung und weitere verkehrliche, betriebliche und wirtschaftliche Gesichtspunkte beachtet werden. Aus diesen genannten Gründen ist der Bau einer Schallschutzwand oftmals unvermeidlich. Die Unterscheidung der Lärmschutzwände erfolgt nach deren Reflexionseigenschaften. So gliedern sich diese in reflektierende, absorbierende und hochabsorbierende Wände. Reflektierende Lärmschutzwände werden an Straßen mit keiner schutzbedürftigen Bebauung, oder Bebauungen außerhalb der Strahlenreichweite eingesetzt. Mithilfe von absorbierenden Schallschutzwänden kann der Reflexion durch hohe Fahrzeugaufbauten entgegengewirkt werden. Hochabsorbierende

Lärmschutzwände sind nur bei hohen Verkehrsaufkommen und nahen Bebauungen, für eine Verringerung der Schallausbreitung notwendig (Bundesministerium für Verkehr, 1990).

2.2.5 Lärmberechnung

Für die Implementierung einer grafischen Lärmsimulation, bedarf es zunächst an einer Orientierung über das Formelwerk. Die 16. Verordnung zur Durchführung des Bundes-Immissionsschutzgesetzes beschreibt das grundsätzliche Vorgehen bei Berechnungen an Verkehrswegen. Diese stützt sich auf die RLS-90 Norm und schreibt vor, diese bei nicht standardmäßigen Voraussetzungen, wie krummen Straßen zur Berechnung zu verwenden (Bundesrepublik Deutschland, 1990).

Da die Implementierung des Tools unter anderem auch nicht standardmäßige Trassen beinhaltet wurde die RLS-90 Norm zur Lärmberechnung verwendet. Die Lärmermittlung an einer gekrümmten Fahrbahn benötigt das Teilstück-Verfahren. Die Norm gibt diese Methode für die Zerlegung einer gekrümmten Fahrbahn vor, damit anschließend Berechnungen an den einzelnen Teilstücken durchgeführt werden können. Die Anzahl der Teilstücke hängt dabei von dem geringstem Abstand s zwischen Immissionsort und Emissionsort ab. Generell lässt sich sagen, dass mit einer hohen Anzahl an Teilstücken ein genaueres Ergebnis erzielt werden kann. Dabei ist zu beachten das die Teilstücklänge maximal 0.5 s betragen darf, da sonst Ungenauigkeiten entstehen können (Bundesministerium für Verkehr, 1990).

Für die einzelnen Teilstücke ergibt sich folgender Mittelungspegel:

$$L_{m,i} = L_{m,E} + D_l + D_s + D_{BM} + D_B \tag{2.1}$$

mit
- $L_{m,i}$: Mittelungspegel eines Teilstückes
- $L_{m,E}$: Emissionspegel
- D_l : Berücksichtigung der Teilstücklänge und Pegeländerung
- D_s : Berücksichtigung des Abstandes und Luftabsorption
- D_{BM} : Boden und Meteorologiedämpfung
- D_B : topografischen und baulichen Gegebenheiten

Die Bestandteile von Formel 2.1 sind getrennt zu berechnen. Es folgt eine nähere Betrachtung der einzelnen Formel-Elemente. Der Emissionspegel $L_{m,E}$ beschreibt sich hierbei durch folgende Formel:

$$L_{m,E} = L_m^{25} + D_v + D_{Stg} + D_E \tag{2.2}$$

mit $L_{m,E}$: Emissionspegel
L_m^{25} : Mittelungspegel (25)
D_v : Korrektur für Höchstgeschwindigkeit
D_{Stg} : Korrektur für Steigung
D_E : Korrektur für Spiegelschallquellen

Der Emissionsort wird dabei in der Mitte des Teilstückes in 0,5 Meter Höhe über dem Fahrstreifen angenommen. Der Mittelungspegel L_m^{25} gilt für einen horizontalen Abstand von 25 Metern, einer Straßenoberfläche aus nicht geriffeltem Gußasphalt, einer zulässigen Höchstgeschwindigkeit von 100 km/h, einer Steigung oder einem Gefälle von maximal 5% und einer freien Schallausbreitung (Bundesministerium für Verkehr, 1990).

Dieser ergibt sich folgendermaßen:

$$L_m^{25} = 37,3 + 10\,lg(M\,(1 + 0,082\,p)) \tag{2.3}$$

mit L_m^{25} : Mittelungspegel (25)
M : Verkehrsstärke
p : LKW-Anteil

Durch die weiteren Komponenten der Formel 2.2 wird der Mittelungspegel L_m^{25} um die oben genannten Parameter angepasst. Da dieses Plugin nur für eine Lärmberechnung in den frühen Leistungsphasen gedacht ist, werden einige Faktoren vernachlässigt. Somit wird eine Minimierung des Rechenaufwandes erzielt und vorerst irrelevante Genauigkeiten können umgangen werden. Dennoch gibt es noch weitere wichtige Elemente in der Formel 2.2, die im Folgenden erläutert werden. Als Erstes folgt eine Korrektur zur Berücksichtigung der Teilstücklänge D_l. Da die Teilstücke nicht zwangsläufig die gleiche Länge besitzen, muss der Emissionspegel um diesen Faktor angepasst werden:

$$D_l = 10\,lg(l) \tag{2.4}$$

mit D_l : Berücksichtigung der Teilstücklänge und Pegeländerung
l : Teilstücklänge

Des Weiteren ist die Berücksichtigung von Abstand und Luftabsorption D_s wichtig. Diese Komponente wird benötigt, um die Veränderung des Pegels durch verschiedene Abstände von Emissionsorten und Immissionsorten miteinzubeziehen. Dieser ergibt sich zu:

$$D_s = 11,2 - 20\,lg(s) - \frac{s}{200} \tag{2.5}$$

mit D_s : Berücksichtigung des Abstandes und der Luftabsorption
s : Abstand zwischen Emissionsort und Immissionsort [m]

Abschließend sind noch die Bestandteile D_{BM} und D_B in der Teilstückberechnung 2.1 zu finden. D_{BM} beschreibt die Boden- und Meteorologiedämpfung. Um den Rechenaufwand in Grenzen zu halten wird dieser Faktor ebenfalls vernachlässigt. Für die Ermittlung, ob eine bauliche Gegebenheit angesichts des Schallschutzes benötigt wird, ist es dennoch erforderlich die Pegeländerung durch topografische und bauliche Gegebenheiten D_B mit einzubeziehen. Diese Komponente ergibt sich folgendermaßen:

$$D_B = D_{refl} - D_z \tag{2.6}$$

mit D_B : Boden- und Meteorologiedämpfung
D_{refl} : Pegelerhöhung durch Mehrfachreflexionen
D_z : Abschirmmaß

Da sich die Pegelerhöhung durch Mehrfachreflexionen nur auf Fahrstreifen zwischen parallelen Wänden bezieht (Straßenschluchten, Troglagen) wird diese zur Vereinfachung nicht berücksichtigt. Das Abschirmmaß D_z lässt sich folgendermaßen beschreiben:

$$D_z = 10\,lg(3 + 80\,z\,K_w) \tag{2.7}$$

mit D_z : Abschirmmaß
z : Schirmwert
K_w : Witterungskorrektur zur Berücksichtigung der Strahlenkrümmung

K_w ergibt sich hierbei folgendermaßen:

$$K_w = e^{-\frac{1}{2000}\sqrt{\frac{A\,B\,s}{2\,z}}} \tag{2.8}$$

mit K_w : Witterungskorrektur zur Berücksichtigung der Strahlenkrümmung
A : Abstand des Emissionsort von der ersten Beugungskante
B : Abstand der letzten Beugungskante vom Immissionsort
z : Schirmwert

Der Schirmwert z ist die Differenz zwischen der Länge des Weges vom Fahrstreifen über die Beugungskanten zum Immissionsort und dem Abstand zwischen Fahrstreifen und Immissionsort, er ergibt sich zu:

$$z = A + B + C - s \tag{2.9}$$

mit z : Schirmwert
A : Abstand des Emissionsort von der ersten Beugungskante
B : Abstand der letzten Beugungskante vom Immissionsort
C : Summe der Abstände zwischen mehreren Beugungskanten

Folgende Grafik ist zur Veranschaulichung der Elemente A, B, C und s da:

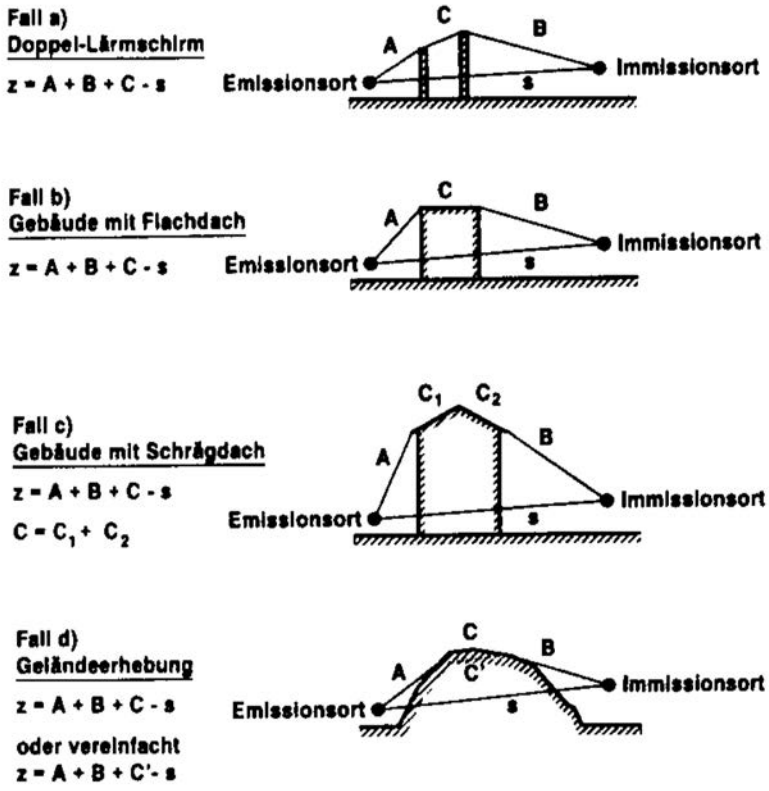


Abbildung 2.5: Schirmwert z bei mehreren Beugungskanten (Bundesministerium für Verkehr, 1990)

Dabei wurde sich, um spätere unnötige Komplikationen zu vermeiden ausschließlich auf den Fall b) bezogen. Mithilfe dieser Gleichung lässt sich die Abschirmung eine Gebäudes oder einer Lärmschutzwand ermitteln. Damit die Lärmberechnung für einen Immissionspunkt abgeschlossen werden kann, müssen die mit der Teilstückformel 2.1 berechneten Elemente logarithmisch addiert werden. Die einzelnen Teilstücke lassen sich zusammenfassen mit:

$$L_m = 10\,lg \sum_i 10^{0,1\,L_{m,i}} \tag{2.10}$$

mit L_m : Mittelungspegel gesamte Strecke
$L_{m,i}$: Mittelungspegel eines Teilstücks i
i : Anzahl der Teilstücke

So lässt sich der Mittelungspegel L_m am Immissionsort für einen einzelnen Punkt anhand des Teilstückverfahrens gemäß der RLS-90 Norm bestimmen. Um das ganze grafisch, mithilfe von Isophonen darzustellen, benötigt man eine gitterförmige Berechnung dieser Schritte für eine große Anzahl an Punkten.

2.3 Building Information Modeling

2.3.1 Grundlagen und Entwicklung von BIM

Allgemein

> *„Unter BIM versteht man ein umfassendes digitales Abbild eines Bauwerks mit großer Informationstiefe. Dazu gehören neben der dreidimensionalen Geometrie der Bauteile vor allem auch nicht-geometrische Zusatzinformationen wie Typinformationen, technische Eigenschaften oder Kosten. Der Begriff Building Information Modelling beschreibt entsprechend den Vorgang zur Erschaffung, Änderung und Verwaltung eines solchen digitalen Bauwerkmodells mithilfe entsprechender Softwarewerkzeuge“* (Borrmann *et al.*, 2015, S. 4)

Der Grundgedanke besteht folglich darin, ein Gebäude mithilfe eines digitalen Modells über den gesamten Lebenszyklus zu beschreiben. Dieses beinhaltet sowohl die dreidimensionale Geometrie, als auch die zugehörige Semantik. Der entscheidende Vorteil von BIM ergibt sich durch die Erhaltung und Wiedernutzung des Modells. Dadurch können fehleranfällige Wiedereingaben von bestehenden Informationen vermieden werden (Borrmann *et al.*, 2015).

Das 3D -Modell bietet eine Möglichkeit zur einfachen Ableitung von 2D -Plänen. Ein großer Vorteil zur reinen 3D -Modellierung besteht darin, dass bauspezifische Objekte bereits bestehen. So müssen beispielsweise Türen oder Fenster nicht neu modelliert werden. Um am Ende eines Prozesses Modelle aus den Plänen abzuleiten, ist es erforderlich, diese Objektgruppen zu verwenden, um geltende Vorschriften und Normen einzuhalten. Auch im Planungsprozess ergeben sich durch Building Information Modeling zahlreiche Vorteile. Sämtliche Ansichten, Grundrisse und Schnitte können aus einem Modell abgeleitet werden und widersprechen sich somit nicht. Des Weiteren können Berechnungen und Simulationen mithilfe der Vielzahl der Informationen aus dem Modell durchgeführt werden (Borrmann *et al.*, 2015).

So lassen sich neben statischen Berechnungen auch Lärmsimulationen durch die enorme Informationstiefe ableiten. Es ist ebenfalls möglich Kollisionskontrollen zwischen den einzelnen Teilmodellen durchzuführen, um einen frühzeitigen Überblick über die Korrektheit und Vollständigkeit des Modells zu bekommen. Darüber hinaus können auch präzise

Abbildung 2.6: Building Information Modelling (Borrmann *et al.*, 2015)

Mengenermittlungen und Prüfungen von gesetzlichen Normen ausgeführt werden. Zusammenfassend entsteht durch den Einsatz von BIM eine Aufwandsverlagerung. Das bedeutet, dass einzelne Prozesse bereits in früheren Phasen bearbeitet werden können. Das ermöglicht eine Anwendung von vielen Simulationswerkzeugen, wie beispielsweise auch innovative Lärmberechnungen, was in späteren Leistungsphasen nicht mehr möglich wäre (Borrmann *et al.*, 2015).

Strategiekonzept

> *„Das Strategiekonzept hat zum Ziel, den bauwerksmodellbasierten Ansatz für die Optimierung der Planung-, Ausführungs- und Bewirtschaftungsprozesse im Bauwesen in Deutschland zu etablieren.“* (Liebich & Gersching, 2010, S. 4)

Dieses Ziel hat in den letzten Jahren eine bedeutende Rolle in der Baubranche bekommen. Wenn es nicht gelingen sollte, besteht die Gefahr, dass die deutsche Bauwirtschaft den Anschluss zu internationalen Planungsbüros verlieren könnte. BIM spielt dabei eine entscheidende Rolle (Liebich & Gersching, 2010).

Als Grundlage für Building Information Modeling sind umfassende Lösungen mit offenen Standards, wie beispielsweise IFC nötig. Die Hauptziele des Strategieprozesses, gemäß Liebich & Gersching (2010) sind:

- wirtschaftliche Vorteile generieren
- effektives Arbeiten mit modernen IT-Werkzeugen
- durchgängige Datennutzung
- besserer Informationsaustausch
- Stärkung der deutschen Bauwirtschaft
- Bessere Informationsübergabe an die öffentliche Hand

Um diese Ziele erreichen zu können, müssen noch einige notwendige Schritte eingeleitet werden. Für alle Projekte müssen allgemeine Vorgaben zur Umsetzung einer BIM-basierten Projektabwicklung konzipiert werden. Des Weiteren ist es erforderlich, die Urheberschaft über das Bauwerkmodell zu klären. Der Einfluss des Modells auf die einzelnen Phasen des Modells muss zunächst einmal betrachtet werden. Daneben spielen auch die organisatorischen Voraussetzungen eine große Rolle. Einzelne Teilhaber müssen für die Teamarbeit mit BIM-Strukturen ausgebildet werden. Es stellt sich ebenfalls die Frage, ob es an Spezialisten bedarf und welche Aufgabenfelder diese zu decken haben (Liebich & Gersching, 2010).

2.3.2 Openstreetmap

Allgemeines

Openstreetmap ist ein Community Projekt, welches 2004 von Steve Coast in Großbritannien in die Wege geleitet wurde. Die Geodaten werden auf streetmap.org veröffentlicht. Ziel der Plattform ist es, die von Nutzern gesammelten Daten zusammenzutragen und anderen Nutzern urheberrechtlich frei zur Verfügung zu stellen. Dabei handelt es sich hauptsächlich um geografische Daten. Der Statistik Report aus dem Jahr 2019 zeigt, dass mittlerweile 7 Milliarden GPS-Punkte hochgeladen wurden und 5 Millionen Nutzer auf Openstreetmap angemeldet sind. Mittlerweile sind mit die meisten Regionen Deutschlands abgedeckt. Dennoch wird weiter an der Vergrößerung der Datenbank gearbeitet (Fossgis e.V., 2019).

Die Verarbeitung der geografischen Information erfolgt mittels Geographic Information Systems (GIS). Die Datenformate und Software sind auf die spezifischen Bedürfnisse der Plattform zugeschnitten. Durch die freie Zugänglichkeit können Karten, mit wenigen Zeilen Programmcode in andere Software eingebunden werden (Ramm & Topf, 2010).

OpenStreetMap und CDP

Die Collaborative Design Plattform CDP bietet eine Einlesemöglichkeit von Openstreetmaps. Diese ist eine semantische Datenschnittstelle als Plangrundlage anhand von GIS-Daten im Tisch integriert. Der Benutzer hat zunächst die Möglichkeit ein bestimmtes Gebiet auszuwählen. Dieses wird durch C++ Implementierungen an den interaktiven Tisch weitergegeben und in eine C# -Library *context* übertragen. In der Klasse *context* lassen sich Daten, wie Gebäude *context.buildings* oder Straßen *context.streets* finden (Schubert, 2012).

2.3.3 Industry Foundation Classes

Allgemeines

Durch die zunehmende Verwendung von BIM -Modellen gestaltete sich eine Problematik: Wie können verschiedene Programme mit verschiedenen Datenformaten an einem Modell arbeiten?

IFC bietet für dieses Problem eine Lösung. Unter Industry Foundation Classes (IFC) versteht man ein standardisiertes Produktdatenmodell für das Bauwesen. Dieses wurde von der buildingSmart International entwickelt. Von vielen BIM-Tools wird der Export und Import von IFC-Files angeboten. Die aktuelle Version ist IFC 4x2 (buildingSmart International, 2019).

Entwicklung von IFC

Industry Foundation Classes IFC wurde ursprünglich, als ein offenes und standardisiertes Datenmodell konzipiert, um eine Verknüpfung zwischen BIM-Software und der Arichtecture, Engineering and Construction (AEC) Industrie zu schaffen. Das Ursprungskonzept wurde 1994 von einem Industrie-Bund entwickelt. Dennoch haben sich die Ziele im Laufe der Jahre verändert. 1994 war IFC ausschließlich dafür gedacht, ein offenes Datenformat zu gestalten, um die Bedürfnisse der BIM -Zusammenarbeit der Industrie zu decken. Trotz der Veröffentlichung von zahlreichen IFC-Versionen ist die tatsächliche Verwendung in der Industrie gering geblieben. Das lag vor allem an der Problematik gesetzlicher Restriktionen. Bedauerlicherweise war der Austausch von BIM-Daten davon abhängig. Die meisten Projekte sind auf spezifischer Software gestützt, woran sich alle Teilhaber richten sollten, trotz des Faktes, dass gleichzeitig an einem offenen Datenaustauschmodell gearbeitet wurde. IFC hätte somit in seinen frühen Jahren das Potential gehabt eine Verknüpfung zwischen den einzelnen Projekt-Phasen in einer zerstückelten Projekt Umgebung zu bilden (Mikael & Arto, 2012).

IFC unterstützte ebenfalls frühzeitig die modellbasierte Konstruktion und hätte somit Kern des Konstruktionsprozesses werden können. Das Potential für hohe Produktivität war enorm.

Dennoch waren die Schwierigkeiten damals zu groß. Die offene Zusammenarbeit in BIM - Lösungen führt zu einer Verschmelzung von Information über Gestaltung, Kosten, Produktion und des allgemeinen Projekts. Dadurch könnte, über die gesamte Lebenszeit eines Gebäudes betrachtet, die Effizienz gesteigert werden und die Redundanz einiger Prozesse vermindert werden. Als maßgebendes Bindeglied für diesen Prozess betrachtet, ist IFC eine der wertvollsten IT-Standardisierungen der gesamten Industrie (Mikael & Arto, 2012).

Insgesamt ist es von großer Bedeutung die Langzeitentwicklung der Industry Foundation Classes zu betrachten und nicht nur den aktuellen Output. Der Entwicklungsprozess hat sich geändert. Die Standardisierung beinhaltet nicht mehr das triviale Übernehmen und Aufbauen auf Funktionen von existierender Software. Ziel ist es viel mehr geworden, die Bedürfnisse der Industrie mit neuen technologischen Entwicklungen zu decken (Mikael & Arto, 2012).

Um den Bedürfnissen der Industrie nachzukommen, ist es ebenfalls wichtig, dass das Lärm-Tool Informationen in ein offenes Datenformat schreiben kann. Dieser Prozess wird im folgenden Abschnitt beschrieben.

IFC-Wall

Bis zum heutigen Stand gibt es noch keine IFC-Gruppierung, in der das spezifische Objekt Lärmschutzwand gespeichert wird. Damit die Position, Höhe und Form dennoch aufgenommen werden können, bietet es sich an die Schallschutzwände unter dem Typ IfcWall einzugliedern (buildingSmart International, 2019b).

Die Wände werden durch die ISO 6707-1 definiert. In den Industry Foundation Classes werden vertikale Konstruktion in zwei Fälle gegliedert. Zum einen gibt es IfcWallStandardCase für alle Wände, deren Breite entlang des Wandpfades gleich bleibt und deren Dicke durch Materialparameter beschrieben werden kann. Eine weitere Option Wände in IFC zu beschreiben ist der IfcWallElementedCase. Dieser ist für alle Wände, die aus untergeordneten Elementen abgeleitet werden und bestimmten Zerlegungsregeln folgen. Das sind auch Wände mit veränderlicher Dicke entlang des Pfades. Des Weiteren werden darin krumme Formen, schräge Wände und Wände die aus Boundary Representation (BREP) Geometrie (Formen werden durch ihre begrenzte Oberfläche definiert) bestehen, beschrieben (buildingSmart International, 2019).

Der Wandtyp und Informationen über Wandstile kann dabei mittels IfcWalltype beschrieben werden. Des Weiteren können dort Attribute, wie Name, Material und weitere Eigenschaften definiert werden. Eine weitere Möglichkeit das Material zu beschreiben ist durch die IFC-Klasse IfcMaterialLayerSet. Vielschichtige Wände werden durch den Bezug auf bestimmte IFC-Materiallayer innerhalb der IfcMaterialLayerSet Umgebung spezifiziert. Materialtypen werden dann durch die IfcWallType Klasse beschrieben, wenn die

Materialwerte typabhängig sind. Die Eigenschaften einer IfcWall werden über IfcPropertySet bestimmt. Für die IfcPropertySet Klasse gibt es ein vordefiniertes IfcPropertySet: *Pset-WallCommon*. Dieses gilt für Wände mit gewöhnlichen Eigenschaften. Die Beschaffenheiten können ebenfalls durch den IfcWallType besetzt werden, wenn diese typabhängig sind. Falls die Eigenschaften mehrfach besetzt wurden werden die des IfcWallTypes überschrieben. Die Mengen, welche sich auf die IFC-Wände beziehen sind in IfcElementQuantity definiert. Diese können nicht in der IfcWallType Klasse belegt werden. Für die Einordnung der IFC -Wand ergeben sich zwei Optionen. Zunächst einmal kann die Wand hierachisch integriert werden. Dabei werden die Objektbeziehungen anhand der Klasse IfcRelContainedInSpartialStructure beschrieben. Diese Möglichkeit wird in den meisten Fällen verwendet. Daneben können die IFC-Wände auch in eine Elementstruktur, mittels IfcRefAggregate eingebettet werden. Die geometrische Definition einer IfcWall wird durch IfcProductDefinitionShape ermöglicht. Dadurch ergeben sich viele verschieden Darstellungsmöglichkeiten. Es bestehen Optionen einer geometrischen, achsbezogenen, oberflächen- oder körperdefinierten Visualisierung. Diese inkludiert Möglichkeiten für geometrische Operationen wie Sweeps, Clipping oder BREP für die Gestaltung. Dabei können die Elemente sowohl in einem lokalen, relativem Koordinatensystem, als auch in einem absoluten System (wenn kein relativer Bezug vorhanden ist) platziert werden. Die Verknüpfung und Verbindung zweier Wand-Geometrien erfolgt mit IfcRelConnectsPathElements. Die Verwendung der gekoppelten Parameter ist dabei sowohl in der übergeordneten IfcWall Klasse, als auch in IfcRelConnectsPathElements definiert (buildingSmart International, 2019).

IFC-LinearPlacement

IfcLinearplacement ist eine Unterklasse des IfcObjectplacement. Es wurde mit der IFC-Version 4x1 2018 veröffentlicht. Damit ist es möglich die Position eines Objektes nicht nur durch x und y- Koordinaten zu definieren, sondern auch entlang eines IfcAlignments. Die Instanz IfcLinearPlacement wird durch die Attribute *PlacmentRelTo*, *Distance*, *Orientation* und *CartesianPosition* beschrieben. Unter *PlacementRelTo* ist die IfcCurve zu der das Objekt bezogen ist festgelegt. Der Abstand und somit die relative Position zur Kurve wird mit IfcDistanceExpression definiert. Wie das platzierte Objekt bezüglich der Kurve ausgerichtet ist bestimmt IfcOrientationExpression. Es kann als zusätzliche Sicherheit ein IfcAxis2Placement3D angegeben werden. Falls ein Programm IfcLinearPlacement nicht unterstützt, wird auf die angegebene Position zurückgegriffen (buildingSmart International, 2019a).

Mithilfe von IfcLinearplacement ist es somit möglich Objekte, wie beispielsweise eine bauliche Schallschutzmaßnahme bezüglich einer Trasse zu platzieren. Das vereinfacht nicht nur die Positionierung, sondern gibt eine zusätzliche Sicherheit über die festgelegte Lage.

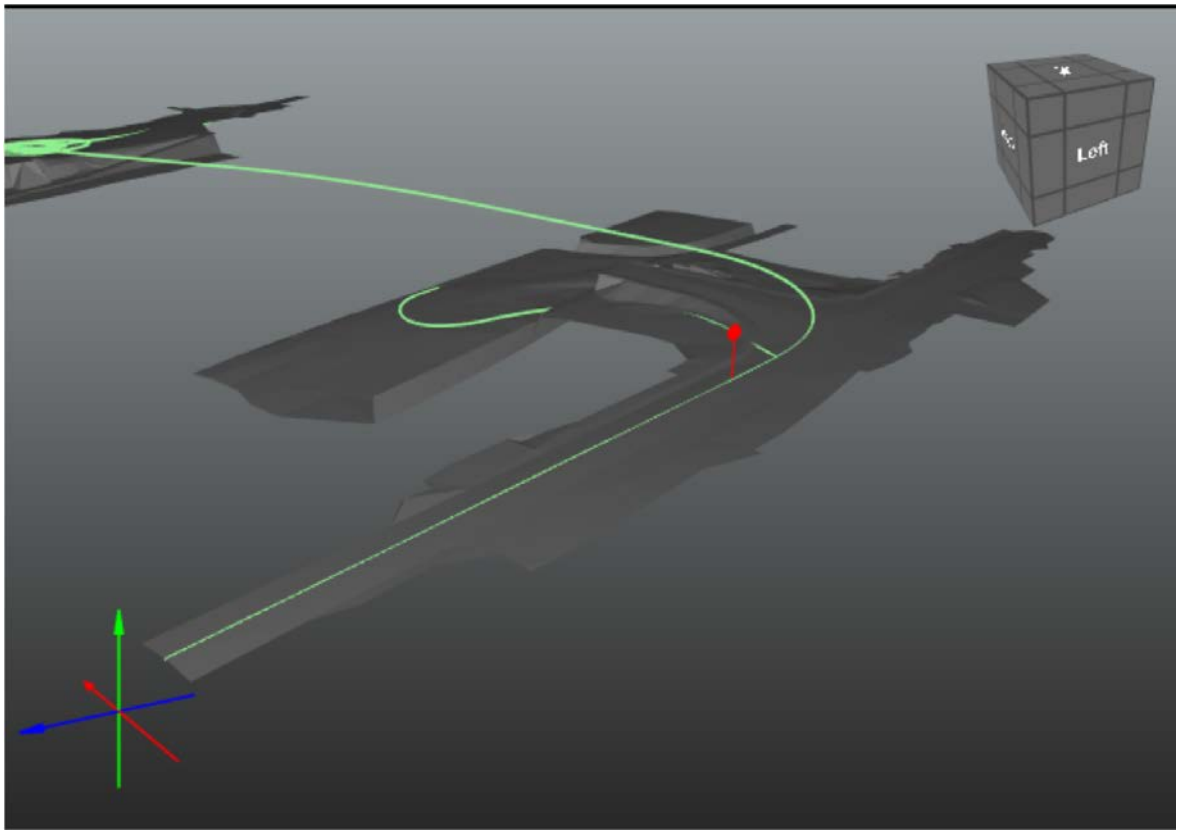


Abbildung 2.7: IfcLinearPlacement: Ein Signalschild entlang einer Trasse positioniert (buildingSmart International, 2019a)

Kapitel 3

Implementierung

3.1 Prozess

Um eine Vorstellungen darüber zu bekommen, wie das Tool funktioniert, wurde eine einfache Prozessstruktur entworfen. Zunächst hat der Nutzer die Option eine Kartenregion auszuwählen, um die Gebäudedaten einzulesen. Diese werden dann an das Plugin übergeben und angezeigt. Mithilfe des Alignmenttools kann der Benutzer dann eine Achse erstellen. Anschließend besteht die Möglichkeit eine Lärmberechnung durchzuführen. Anhand des erstellten Bildes kann der Benutzer eine geeignete Position der Lärmschutzwand auswählen. Mithilfe von Blöcken wird der bauliche Schallschutz auf dem Tisch platziert und eine neue Simulation gestartet. Es besteht jederzeit die Möglichkeit die Position der Lärmschutzwand oder den Verlauf der Achse zu ändern. Wenn der Benutzer mit seinen Ergebnissen zufrieden ist, können diese in einer IFC-Datei gespeichert werden. Die ermittelte Lärmausbreitung kann ebenfalls als Screenshot ausgegeben werden.

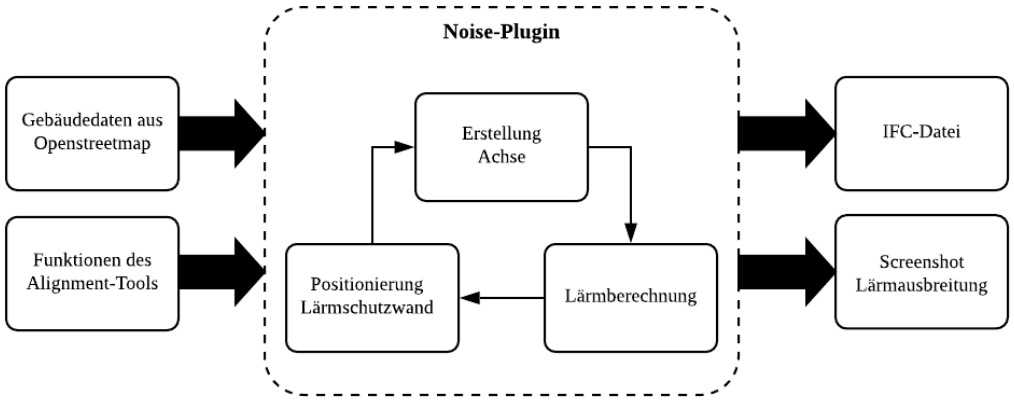


Abbildung 3.1: Prozess

3.2 Klassenstruktur

Der Grundgedanke bei dem Entwurf des Tools bestand darin eine möglichst stabile Struktur zu erstellen. Dabei galt es zwei wichtige Prinzipien des objektorientierte Programmierens zu berücksichtigen. Zum einen ist das die Vererbung, wobei es eine übergeordnete Klasse gibt der sich alle weiteren unterzuordnen haben. Dadurch können wiederholte Implementierungen einzelner Methoden vermieden werden. Zum anderen wurde das Prinzip der Verkapselung berücksichtigt. Dieses sagt aus, dass jede Klasse für nur seine eigenen Methoden und Parameter zuständig ist. Dieses Prinzip fördert ebenfalls die Stabilität eines Programms, insbesondere weil dadurch fehlerhafte oder falsch interpretierte Aktionen im Voraus vermieden werden können (Microsoft, 2019).

Das Plugin besteht aus einer Hauptklasse die viele weitere Klassen beinhaltet. Die Elternklasse aller untergeordneten Klassen ist *NoiseSimulation*. Diese Klasse tritt nur im Hauptprogramm auf. Die Unterklassen wurden alle mit *Noise-* gekennzeichnet, um eine Zugehörigkeit zuzuweisen. Die einzelnen Members der Klasse sind *Truckshare*, *AverageCarPerHour*, *DayMode* und *BuildingMode*. *Truckshare* beschreibt den LKW-Anteil einer Simulation und ist als *double*-Typ deklariert. *AverageCarPerHour* ist der DTV und ebenfalls ein *double*-Wert. *DayMode* und *BuildingMode* sind *booleans* und entscheiden, ob die Simulation bei Tag oder Nacht durchgeführt wird und ob Gebäude in die Berechnungen miteibezogen werden. Des Weiteren beinhaltet die Hauptklasse eine Liste an *NoisePoints*, *CurvePoints*, *NoiseBarriers* und *Buildings*. Diese Klassen werden im folgenden Kapitel genauer beschrieben und erläutert.

NoiseRender ist ebenfalls ein Teil der *NoiseSimulation*. Darin werden alle bestehenden Objekte der *NoiseSimulation*-Klasse übernommen und auf dem interaktiven Tisch angezeigt. Eine weitere Unterklasse von *NoiseSimulation* ist *NoiseFunction*. Diese Klasse beinhaltet alle für die Lärmberechnung relevanten Funktionen, hat aber selber keine Members. Stark daran angelehnt ist die *NoiseAuxiliaryFunction* Klasse, die alle generellen Hilfsfunktionen beinhaltet. Im folgenden werden nun die einzelnen Klassen und deren Methoden genauer erläutert.

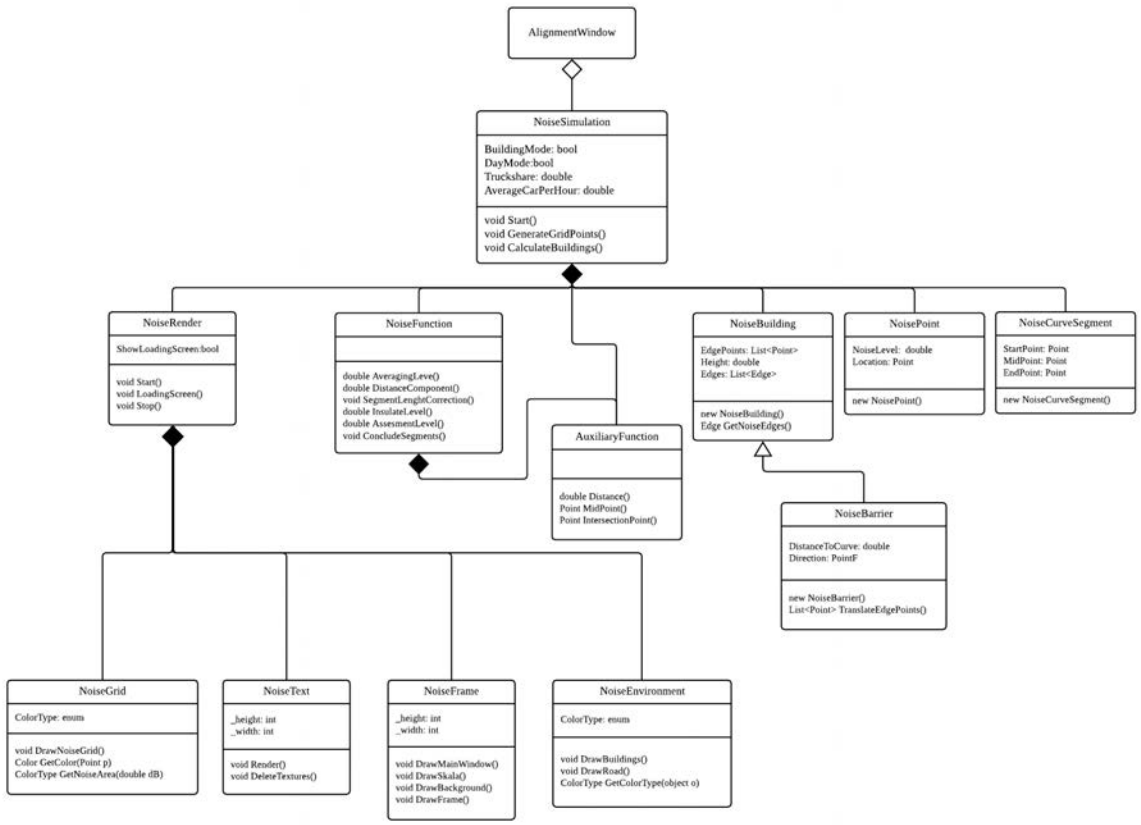


Abbildung 3.2: UML-Diagramm des Tools

3.3 Hilfsfunktionen

Um Lärmberechnungen durchzuführen bedarf es an verschiedenen Hilfsfunktionen. So ist es oftmals nötig den Abstand zwischen zwei Punkten zu bestimmen oder den Mittelwert zweier Werte zu ermitteln. Um sich nicht übermäßig zu wiederholen wurde eine Klasse AuxiliaryFunction erstellt. Diese beinhaltet sämtliche Hilfsfunktionen, die im folgenden näher beschrieben werden.

Abstand zwischen zwei Punkten

Der Abstand zweier Punkte lässt sich durch die Euklidische Norm bestimmen. Dafür wurde die Funktion *Distance* implementiert. Die Funktion hat die Möglichkeit als Eingabeparameter die x- und y-Koordinaten des ersten Punktes und zweiten Punktes zu übernehmen (x1,y1,x2,y2). Des Weiteren existiert eine Überladung der Funktion, welche die Möglichkeit bietet zwei Punkte der C#-Klasse *PointF* einzulesen.

Mittelpunkt zwischen zwei Punkten

Der Mittelpunkt zweier Geraden lässt sich mit Hilfe einfacher Mittelwertfunktionen bestimmen. Es bestehen verschiedene überladbare Operatoren der Funktion, so dass es möglich ist, sowohl einen Punkt mit *float*-Werten als Rückgabeparameter zu erhalten, als auch mit sämtlichen anderen Datentypen. So gestaltet sich die Verwendung der Funktion als flexibel.

Schnittpunkt zwischen zwei Strecken

Der Schnittpunkt zweier Geraden lässt sich anhand von mathematischen Formeln bestimmen. Dabei wurde sich, für einen optimalen und schnellen Berechnungsprozess vor allem an den Vorgehensweisen von LaMothe (2002) orientiert. Zunächst müssen die Mittelpunkte zweier Werte bestimmt werden. Dann kann mithilfe eines Vektoprodukts auf die Mittelpunkte geschlossen werden. Wenn ein Schnittpunkt gefunden wurde, wird dieser zurückgegeben. Des Weiteren muss sich dieser nicht nur auf der Gerade, sondern auch zwischen Start- und Endpunkt der Strecke befinden. Für das Vorgehen wurde die Funktion *FindIntersectionPoints* mit einem Nullable-Typen als Rückgabeparameter erstellt. Falls ein Schnittpunkt gefunden wurde wird dieser zurückgegeben, ansonsten wird der Wert als null deklariert.

Schnittpunkte zwischen Gebäude und Linie von Emissions- und Immissionspunkt

Um die Berechnung zu beschleunigen wurden folgende Vereinfachungen angenommen. Ein Gebäude besteht aus vier Kanten und hat entweder zwei Kantenschnittpunkte zwischen Immissionsort und Emissionsort, oder gar keine. Mithilfe dieser Vereinfachungen lassen sich die Parameter der RLS-90 Norm bestimmen.

Die Ermittlung der zwei Schnittpunkte erfolgt mit den oben genannten Funktionen. Des weiteren gibt es eine Methode *FindBuildingIntersectionPoints*, die prüft, ob zwei Kanten des Gebäude jeweils einen Schnittpunkt besitzen. Wenn dies der Fall sein sollte, werden an die NoiseSimulation-Klasse die entsprechenden Punkte zurückgegeben. Mit diesem Prozess kann schnell getestet werden, ob Schnittpunkte mit den Gebäudeachsen existieren, oder nicht.

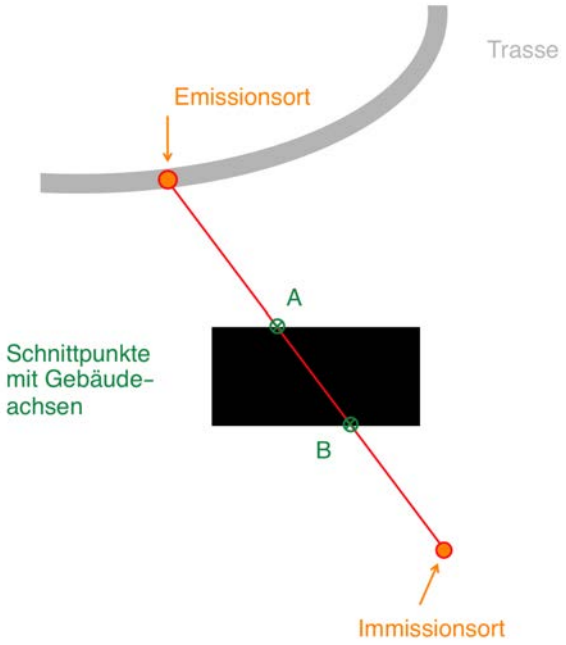


Abbildung 3.3: Darstellung Gebäudeschnittpunkte

3.4 Initialisierung

3.4.1 Erstellen eines Punktegitters

Zunächst einmal stellt sich die Frage wie man eine Berechnung an mehreren Messpunkten gestalten kann. Mithilfe eines Punktegitters kann eine durchgehend gleichmäßige, flächenübergreifende Berechnung erzielt werden. Um den Rechenaufwand nicht allzu groß zu gestalten erschien die Wahl eines 100 x 100 Punktegitters zunächst sinnvoll. Dabei sind die Abstände zwischen den Punkten, entsprechend Bayrisches Landesamt für Umwelt (2019), jeweils auf 10 m gesetzt.

Für die einzelnen Gitterpunkte wurde eine Klasse *NoisePoint* erstellt. Jeder *NoisePoint* besitzt ein Member *Location*. Diese besteht aus einer internen C# -Klasse *Point*. *Point* hat die Members *X*, *Y* und *isEmpty*, er beschreibt somit einen einfachen 2D-Punkt. Die weiteren Members von *NoisePoint* sind *NoiseLevel* und *DistanceToCurve*. *NoiseLevel* besteht aus einem primitiven Typen *double* und gibt den Lärmpegel eines Punktes an. *DistanceToCurve* beschreibt eine Liste an *double*-Werten, worin alle Abstände zu den einzelnen Kurvenpunkten gespeichert sind. Somit wird nur eine einmalige Berechnung der Abstände zwischen den Trassenpunkten, welche sich auch als Emissionspunkte gestalten und den Gitterpunkten, auch Immissionspunkten, benötigt. Folglich können bessere Laufzeitergebnisse erreicht werden. Es wurden ebenfalls verschiedene Konstruktoren erstellt um einen *NoisePoint* zu generieren. Da-

mit kann zunächst nur die Position eines Punktes gespeichert werden, ohne dass ein dB-Wert erforderlich ist. In dem unteren Code Ausschnitt ist die Klasse *NoisePoint* dargestellt.

NoisePoint

```
1  public class NoisePoint
2  {
3      public Point Location { get; set; }
4      public double NoiseLevel { get; set; }
5      public List<double> DistanceToCurve { get; set; }
6
7      public NoisePoint(Point location)
8      {
9          this.Location = location;
10     }
11
12     public NoisePoint(Point location, double noiseLevel)
13     {
14         this.Location = location;
15         this.NoiseLevel = noiseLevel;
16     }
17
18     public NoisePoint(Point location, double noiseLevel, List<double>
           distance)
19     {
20         this.Location = location;
21         this.NoiseLevel = noiseLevel;
22         this.DistanceToCurve = distance;
23     }
24 }
```

Für die Erstellung eines Punktegitters werden logischerweise mehrere *NoisePoints* benötigt. Die *NoisePoints* werden in der Klasse *NoiseSimulation* in einer Liste an *NoisePoints* gespeichert. Das hat den Vorteil, das alle Members der C# -Liste auf die einzelnen Objekte angewandt werden können. Jede 10 Meter soll ein Punkt entstehen, deswegen werden die Iterationsparameter i und j in jeder Schleife um den Wert 10 erhöht. Wenn diese den maximalen Wert der äußeren Schleife von beispielsweise 1000 erreichen, ist eine Menge von 100 x 100 Punkten generiert. Die Befüllung der Liste mit einzelnen Werten ergibt sich folgendermaßen:

Erstellung von NoisePoints

```
1  ...
2  for (int i = 0; i < gridLength; i+=10)
3  {
4      for (int j = 0; j < gridWidth; j+=10)
5      {
```

```
6             NoisePoints.Add(new NoisePoint(new Point(i,j)));
7         }
8     }
9     ...
```

3.4.2 Einlesen der Trasse

Für die Berechnung nach dem Teilstückverfahren ist es wichtig die Trasse in einzelne Segmente zu zerlegen. Dafür wurde die Klasse *NoiseCurveSegment* erstellt. Diese beinhaltet die Members *StartPoint*, *MidPoint*, *EndPoint* und *SegmentLength*. Die Position eines Teilstückes orientiert sich an dem Startpunkt eines Elements und ist deswegen unabdingbar. Die Orte der Mittelpunkte der einzelnen Elemente werden unter *MidPoint* gespeichert. Diese sind später ausschlaggebend für die Berechnung, da sie unter anderem auch die einzelnen Emissionsorte der Trasse beschreiben. Da der Mittelpunkt zunächst berechnet werden muss, gibt es keinen Konstruktor für dieses *Member*. Des Weiteren gibt es noch *SegmentLength*, welche den Abstand zum Endpunkt eines Elements angibt. Dieser Abstand entspricht auch der Länge eines Segmentes. Im folgenden Code-Beispiel wird ein Ausschnitt der Klasse NoiseCurvePoint veranschaulicht:

NoiseCurvePoint

```
1  public class NoiseCurveSegment
2  {
3      public Point StartPoint { get; set; }
4      public Point MidPoint { get; set; }
5      public Point EndPoint { get; set; }
6      public double SegmentLength { get; set; }
7
8      public NoiseCurveSegment(Point location)
9      {
10         this.StartPoint = location;
11     }
12
13     public NoiseCurveSegment(Point location, double segmentLength)
14     {
15         this.StartPoint = location;
16         this.SegmentLength = segmentLength;
17     }
18 }
```

Mithilfe einer Liste an *CurveSegements* werden diese Kurvenpunkte aus der Klasse Alignment übertragen. Die vollständige Trasse besteht aus Kurvenelementen und Geradenelementen. Die Kurven-Segmente sind jeweils durch 50 Punkte, beliebig erweiterbar, beschrieben. Die Gera-

den bestehen aus einem Start- und einem Endpunkt. Um aus den Segmenten eine gesamte Kurve zu generieren, war es zunächst erforderlich weitere Punkte zu erstellen. Mithilfe von einzelnen Methoden werden diese generiert und gespeichert. Schlussendlich kann die komplette Trasse in gleichmäßige Segmente zerteilt werden. Anschließend können Berechnungen durchgeführt werden.

3.4.3 Einlesen von Gebäuden

Um die Gebäude übersichtlich einzuordnen, wurde eine Klasse *NoiseBuilding* generiert. Diese beschreibt ein durch OpenStreetMap eingelesenes Gebäude. Ein Gebäude gestaltet sich durch folgenden Members: Einer Liste an Eckpunkte *EdgePoints*, der Höhe *Height* und den einzelnen Kanten *Edges*. *EdgesPoints* sind Punkte der Klasse *NoiseBuildingPoint*. Diese Klasse hat gleichermaßen wie *NoisePoint*, eine Location als C#-Point und zusätzlich eine Höhe *Height* als *double*-Wert. Im folgenden Codesegment wird die Klasse *NoiseBuildingPoint* gezeigt:

NoiseBuildingPoint

```
1  public class NoiseBuildingPoint
2  {
3      public Point Location { get; set; }
4      public double Height { get; set; }
5
6      public NoiseBuildingPoint(Point location)
7      {
8          Location = location;
9      }
10
11     public NoiseBuildingPoint(Point location, double height)
12     {
13         Location = location;
14         Height = height;
15     }
16 }
```

Der Parameter *Height* der Klasse *NoiseBuilding* ist eine *double*-Größe und beschreibt die Gebäudehöhe. Ein Gebäude besteht aus mehreren Kanten. Dafür erschien die Erstellung einer Liste *Edges* mit *NoiseEdges* sinnvoll. *NoiseEdge* ist ebenfalls eine selbst erstellte Klasse. Diese hat die Members *StartPoint*, *EndPoint* und eine Funktion *isOnEdge*. Folglich kann eine Kante mit einem Startpunkt und einem Endpunkt eingelesen werden und es kann überprüft werden, ob ein Punkt auf einer Kante liegt oder nicht.

NoiseEdge

```
1  public class NoiseEdge
```

```
2  {
3      public Point StartPoint { get; set; }
4      public Point EndPoint { get; set; }
5
6      public NoiseEdge(Point startPoint, Point endPoint)
7      {
8          this.StartPoint = startPoint;
9          this.EndPoint = endPoint;
10     }
11
12     public NoiseEdge(PointF stPointF, Point endPoint)
13     {
14         this.StartPoint = Point.Round(stPointF);
15         this.EndPoint = endPoint;
16     }
17
18     public bool isOnEdge(Point p)
19     {
20         if ((StartPoint.X <= p.X && p.X <= EndPoint.X) && (StartPoint.Y
               <= p.Y && p.Y <= EndPoint.Y))
21             return true;
22         else if ((EndPoint.X <= p.X && p.X <= StartPoint.X) && (EndPoint.
               Y <= p.Y && p.Y <= StartPoint.Y))
23             return true;
24         else
25             return false;
26     }
27 }
```

Aus allen oben gezeigten Klassen wurde die Klasse NoiseBuilding erstellt. Für die Erstellung eines Gebäudes wird eine Liste an Eckpunkten und eine Höhe benötigt. Anhand dieser Informationen werden die Kanten des Gebäudes berechnet. Dafür wurde die Methode *CalculateEdges* implementiert. Diese findet und erstellt mithilfe der Eckpunkte die einzelnen Gebäudekanten und gibt diese in einer Liste zurück. Um die Verschachtelung der einzelnen Klassen zu erhalten, wurde die Funktion mit dem Schlüsselwort *private* deklariert. Nur die Klasse Gebäude an sich soll die Möglichkeit haben, ihre eigenen Kanten zu definieren. Die Klasse *NoiseBuilding* ergibt sich somit folgendermaßen:

NoiseBuilding

```
1  public class NoiseBuilding
2  {
3      public List<NoiseBuildingPoint> EdgePoints { get; set; }
4      public double Height { get; set; }
5      public List<NoiseEdge> Edges { get; set; }
```

```
6
7       public NoiseBuilding(List<NoiseBuildingPoint> edgePoints, double
            height)
8       {
9           this.EdgePoints = edgePoints;
10          this.Height = height;
11          this.Edges = CalculateEdges();
12      }
13
14      private List<NoiseEdge> CalculateEdges()
15      {
16          List<NoiseEdge> edges = new List<NoiseEdge>();
17          int nCount = 0;
18          foreach (NoiseBuildingPoint edgePoint in EdgePoints)
19          {
20              NoiseEdge edge = new NoiseEdge(EdgePoints.ElementAt(nCount).
                    Location, EdgePoints.ElementAt(nCount + 1).Location);
21              edges.Add(edge);
22              nCount++;
23              if (nCount == EdgePoints.Count - 1)
24              {
25                  edge = new NoiseEdge(EdgePoints.ElementAt(nCount).
                        Location, EdgePoints.ElementAt(0).Location);
26                  edges.Add(edge);
27                  break;
28              }
29          }
30          return edges;
31      }
32  }
```

Die Gebäude werden anschließend in der NoiseSimulation Klasse in einer Liste NoiseBuildings gespeichert. Die Gebäudeinformationen wurden in C++ über OpenStreetMap eingelesen und sind in *context.buildings* zu finden. Mit einigen Hilfsfunktionen können diese in *NoiseBuildings* umgewandelt werden. Das hat den Vorteil, dass man auf zusätzliche Parameter, wie Kanten und Kantenpunkte zugreifen kann. Der Einleseprozess erfolgt beim Starten des Plugins.

3.4.4 Einlesen der Blöcke als Lärmschutzwände

Wie unter Kapitel 2.1.1 beschrieben hat der CDP-Table auch die Möglichkeit Objekte, welche auf dem Tisch platziert werden einzulesen. Diese werden unter *context.buildings* jeweils als letztes Element der Liste gespeichert. Für die Lärmschutzwände (Blöcke) wurde ebenfalls eine eigene Klasse *NoiseBarrier* erstellt. Diese bekommt alle Funktionen und Members von

NoiseBuilding vererbt. Zusätzlich hat *NoiseBarrier* noch die Methode *TranslateEdgePoints*, was eine Verschiebung an die richtige Position ermöglicht. Des Weiteren wird noch ein Richtungsvektor *Direction* als *PointF* und die kürzeste Distanz zur Achse *DistanceToCurve* als *double*-Wert gespeichert. Damit wird die spätere IFC-Speicherung ermöglicht.

Mithilfe der *Override* Funktionen *onBuildingUp* und *onBuildingDown* kann auf die Blöcke zugegriffen werden. Die einzelnen Objekte werden unter einer Liste *NoiseBarriers* in der *NoiseSimulation* Klasse gespeichert. Der Einleseprozess sieht somit folgendermaßen aus:

Einlesen der Blöcke

```
1 Building building = context.buildings.Last();
2 NoiseBarrier noiseBarrier = new NoiseBarrier(building.vertices2D,
      building.height, HorWindowCornerX, HorWindowCornerY);
3 NoiseSimulation.NoiseBarriers.Add(noiseBarrier);
```

Wenn ein Block platziert wird, wird ebenfalls eine neue Berechnung gestartet, damit man unmittelbar die Veränderungen erkennen kann. Wird der Block wieder entfernt, löscht sich das entsprechende Element wieder aus der Liste und das Lärmbild ändert sich erneut.

3.5 Berechnung der Lärmpunkte

3.5.1 Implementierung der Formelsammlung

Alle verwendeten Formeln sind aus der RLS-90 und wurden unter Kapitel 2.2.5 näher beschrieben. Um diese zu gruppieren wurde die Klasse *NoiseFunction* erstellt. Diese Klasse beinhaltet alle für die Lärmberechnung relevanten Formeln. Um präzise Ergebnisse zu erhalten, gibt jede Funktion einen *double*-Wert zurück. Alle Methoden wurden als *public* und *static* definiert, um einen Zugriff außerhalb der Klasse zu erzielen, ohne vorher eine Instanz zu erstellen. Im folgenden werden die einzelnen Methoden der *NoiseFunction* Klasse kurz beschrieben. Für den Anfang jeder Berechnung benötigt es einen guten Startwert, welcher in der Norm als Mittellungspegel 25 zu finden ist. Dieser wurde als *Averaginglevel* definiert und hat als Eingabeparameter den LKW-Anteil *truckShare* und den DTV.

AveragingLevel

```
1 public static double AveragingLevel(double dtv, double truckShare)
2 {
3     return 37.3 + 10*Math.Log10(0.06 * dtv * (1+0.082*truckShare));
4 }
```

Im Teilstückverfahren treten viele Faktoren zur Reduzierung dieses Parameters auf. Zur Vollständigkeit werden die Parameter Berücksichtigung der Teilstücklänge, Berücksichtigung des Abstandes und der Luftabsorption und die Pegeländerung durch topografische und bauliche Gegebenheiten kurz dargestellt. Eine detaillierte Beschreibung der genannten Parameter ist unter Kapitel 2.2.5 zu finden. Die einzelnen Formeln wurde wie folgt implementiert. Die Berücksichtigung der Teilstücklänge ist mit *SegmentLengthCorrection* definiert und hat die Länge eines Teilstücks als Eingabeparameter.

SegmentLengthCorrection

```
1 public static double SegmentLengthCorrection (double l)
2 {
3     return 10 * Math.Log10(l);
4 }
```

Die Betrachtung des Abstandes und Luftabsorption ist als *DistanceComponent* definiert und hat den Abstand zwischen Emissionsort und Immissionsort als Eingabeparameter.

DistanceComponent

```
1 public static double DistanceComponent(double s)
2 {
3     return 11.2 - 20 * (Math.Log10(s)) - (s / 200);
4 }
```

Falls Gebäude oder Lärmschutzmaßnahmen in die Berechnungen miteinbezogen wurden, folgt ein Korrektur gemäß RLS-90. Der Parameter wurde als *InsulateLevel* definiert und hat als Eingabewerte zwei Schnittpunkte mit den Gebäudekanten, die Gebäudehöhe und den Abstand s zwischen Emissionsort und Immissionsort. Die einzelnen Bestandteile der Formel wurden unter 2.2.5 tiefgehend erläutert.

InsulateLevel

```
1 public static double InsulateLevel(Point p1, Point p2, double height,
      double s)
2 {
3     double a = _auxiliaryFunction.Distance(p1.X, height, p2.Y, height);
4     double b = _auxiliaryFunction.Distance(p2.X, height, p2.Y, height);
5     double c = _auxiliaryFunction.Distance(p1.X, p2.X, p1.Y, p2.Y);
6     double z = a + b + c - s;
7     double kw = Math.Exp((-1 / 2000f) * Math.Sqrt((a * b * s) / 2 * z));
8     return 10 * Math.Log10(3 + 80 * z * kw);
9 }
```

Diese Parameter lassen sich zu einem Beurteilungspegel eines Teilstückes zusammenfassen. Mit *AssessmentLevel* wurde die einzelnen Werte wie folgt integriert.

AssessmentLevel

```
1 public static double AssessmentLevel (double L_me,double D_l, double D_s,
      double D_z)
2 {
3     return L_me + D_l + D_s - D_z;
4 }
```

Die Pegel der Teilstücke lassen sich mit folgender Funktion zusammenfassen:

ConcludeSegment

```
1 public static double ConcludeSegment(double L_mi)
2 {
3     return Math.Pow(10, (0.1 * L_mi));
4 }
```

3.5.2 Berechnung der einzelnen Punkte

Die Berechnung erfolgt gemäß der in Kapitel 2.2.5 beschriebenen Berechnung. Der einzige Unterschied liegt darin, dass das Berechnungsverfahren nun für mehrere Punkte angewandt wird. Die einzelnen Immissionspunkte sind dabei in der Liste NoisePoints gespeichert. Zuerst muss der Mittelungspegel 25 bestimmt werden. Dieser lässt sich entsprechend 2.2.5 bestimmen. Anschließend werden, mithilfe von zwei *foreach*-Schleifen, alle Werte berechnet.

Die erste Schleife besteht aus den Gitterpunkten, die zweite aus den Kurvensegmenten. Jeder Gitterpunkt wird mit einem Kurvenpunkt assoziiert, wodurch sich die einzelnen dB-Werte bilden. Falls ein Gebäude zwischen Emissionsort und Immissionsort liegt, wird dieses erkannt. Darauf folgend ändern sich die Parameter, unter der Verwendung der beschriebenen Hilfsfunktionen und des Lärmberechnungsverfahren, entsprechend ab. Diese werden dann an die einzelnen *NoisePoint* Objekte übergeben. Die einzelnen Distanzen zwischen den Gitterpunkten und den Kurvenpunkten werden ebenfalls als Abstand zwischen Immissionsort und Emissionsort gespeichert, um bei einer zukünftigen Berechnung bessere Laufzeiten zu erzielen.

3.6 Render-Prozess

3.6.1 Allgemein

Das Darstellen der einzelnen Ergebnisse erfolgt auf dem Tisch mit Open GL. Das Paket ist dafür konzipiert, visuelle Formen einfach zu gestalten. Dafür gibt es folgendes Vorgehen: Jeder Zeichenprozess wird mit einer *Begin* Methode gestartet und mit einer *End* Funktion beendet. Dabei besitzt die *Begin* Funktion einen *Enum*-Eingabeparameter, der die zu zeichnende Form

vorgibt. Einfache Figuren, wie Linien Dreiecke oder Polygone sind in dem *Enum PrimitiveType* enthalten. Um beispielsweise Punkte zu zeichnen, muss dieser mit *PrimitiveType.Points* in die *Begin* Methode übergeben werden. Die einzelnen Koordinaten-Punkte werden im zweidimensionalen Raum mit dem Objekt *Vertex2* festgelegt. Diese Klasse beinhaltet verschiedene statische Konstruktoren für einen Punkt. Das bedeutet, dass es nicht erforderlich ist, eine eigene Instanz des Objektes *Vertex2* zu kreieren und dass alle numerischen Datentypen, wie beispielsweise *double*, *float* oder *int* eingelesen werden können. Am Ende eines Zeichenprozesses folgt die *End* Methode. Wenn diese Methode übergeben wurde, registriert der Compiler, welche Knotenelemente zu einer Figur gehören (Silicon Graphics, 2019).

Das Open Gl Paket bietet viele weitere grafische Präsentationsmöglichkeiten. Eine in dem Plugin oft verwendete Option ist die Farbgestaltung. Mittels *Color3* können alle Farbtöne angezeigt werden. Diese Methode bietet als Inputmöglichkeit einen Farbtyp von *System.Drawing* zu verwenden. Das vereinfacht einen schnellen Farbwechsel. Neben der Änderung von Farben gibt es noch weitere Anpassungsmöglichkeiten. Um bestimmte Figuren zu zeichnen ist es oftmals wichtig deren Punktgröße und Linienstärke zu bestimmen. Dafür gibt es die statischen Methoden *PointSize* und *LineWidth*. Um eine Änderung vorzunehmen, müssen die Funktionen vor der *Begin* Methode aufgerufen werden. Diese haben einen Eingabe Parameter mit dem Typ *float*, was eine präzise Größengestaltung ermöglicht. Zum Verständnis folgt ein Beispiel, in dem ein blauer Punkt an der Stelle 10,10 mit der Dicke 10 gezeichnet wird (Silicon Graphics, 2019).

Open GL Beispiel

```
1 public void DrawSimplePoint()
2 {
3     GL.PointSize(10);
4     GL.Begin(PrimitiveType.Points);
5     GL.Color3(System.Drawing.Color.Blue);
6     GL.Vertex2(10,10);
7     GL.End();
8 }
```

3.6.2 Struktur

Für den gesamten Renderprozess sind viele verschiedene Teilprozesse erforderlich. Um diese sinnvoll zu gliedern wurde eine hierachische Struktur erstellt. Die übergeordnete Klasse der RenderKlassen ist die *NoiseSimulation* Klasse. In dieser sollen die Zeichenprozesse einmalig aufgerufen werden. Für die Handhabung der einzelnen Renderprozesse ist nur die *NoiseRender* Klasse zuständig. Diese beinhaltet und steuert sämtliche Unterklassen. Die Unterklassen

haben stets eine Aufgabe und sind weitgehend unabhängig voneinander. Im Folgenden werden die einzelnen Klassen, die für das Anzeigen und Darstellen zuständig sind näher beschrieben.

3.6.3 NoiseRender

Die Klassen *NoiseRender* ist der Kopf aller Render Klassen. Members der Klasse sind *NoiseGrid*, *NoiseFrame*, *NoiseEnvironment* und *NoiseLoadingScreen*. Diese beinhalten die einzelnen Teilaufgaben, wie Zeichnen der Trasse oder Zeigen eines Ladebildschirms. Die Klasse *NoiseRender* verfügt lediglich über zwei Methoden: *Start* und *Stop*. In der *Start* Methode werden alle Zeichenprozesse gestartet und in der *Stop* Methode beendet. Die Funktionen haben weder Eingabeparameter, noch Rückgabewerte, was den Umgang in anderen Klassen vereinfachen soll. Die *Start* Funktion gliedert sich folgendermaßen.

Start des Renderprozesses

```
1  public void Start()
2  {
3      _frame.DrawMainWindow();
4      _frame.DrawBackground();
5      _grid.DrawNoiseGrid();
6      _environment.DrawBuildings();
7      _environment.DrawRoad();
8      _frame.DrawSkala();
9      _frame.DrawRightFrame();
10     if (ClickPointF != null)
11         _grid.RenderCurrentNoise(ClickPointF.Value);
12     _frame.DrawLeftFrame();
13 }
```

Diese Funktion ist dafür da, die einzelnen Prozesse der Teilklassen zu starten, die *Stop* Methode hingegen ist für das Beenden dieser Prozesse zuständig.

3.6.4 Zeichnen der Umgebung

Für das Zeichnen der Umgebung wurde die Klasse *NoiseEnvironment* erstellt. Darin sind die Funktionen *DrawRoad*, *DrawBuildings* und *GetColor* definiert. *DrawRoad* und *DrawBuildings* sind Funktionen ohne Eingabeparameter und Rückgabewerte, um das Aufrufen in *NoiseRender* zu vereinfachen. Beide Methoden sind ähnlich aufgebaut. Als erstes wird überprüft, ob die Objekte existieren. Dann wird eine Linienstärke und eine Farbe definiert. Die Farbe der einzelnen Objekte wird mithilfe von *GetColor* und einem *Enum ColorType* ermittelt. Die

Funktion bekommt als Eingabeparameter einen Typ, wie beispielsweise Gebäude oder Straße übergeben und gibt die entsprechende Farbe zurück. Mithilfe von *foreach-Loops* werden die einzelnen Elemente ausgelesen und an *GL* übergeben. In der Straßen Funktion werden die Startpunkte der Segmente ausgelesen und mit einer Linie verbunden. Bei den Gebäuden werden die Eckpunkte ausgelesen und mithilfe des einfachen Typs *PrimitiveType.Polygon* gezeichnet. Die beiden Funktionen ergeben sich wie folgt:

Zeichnen der Gebäude und des Geländes

```
1  public void DrawBuildings()
2  {
3      if (NoiseSimulation.NoiseBuildings != null)
4      {
5          foreach (NoiseBuilding building in NoiseSimulation.NoiseBuildings
               )
6          {
7              GL.Begin(PrimitiveType.Polygon);
8              GL.Color3(GetColor(ColorType.Building));
9
10             foreach (NoiseBuildingPoint buildingPoint in building.
                   EdgePoints)
11             {
12                 GL.Vertex2(buildingPoint.Location.X, buildingPoint.
                       Location.Y);
13             }
14
15             GL.End();
16         }
17     }
18 }
19
20 public void DrawRoad()
21 {
22     if (NoiseSimulation.CurveSegments != null)
23     {
24         GL.LineWidth(7);
25         GL.Begin(PrimitiveType.LineStrip);
26         GL.Color3(GetColor(ColorType.Street));
27         foreach (NoiseCurveSegment curveSegment in NoiseSimulation.
               CurveSegments)
28         {
29             GL.Vertex2(curveSegment.StartPoint.X, curveSegment.StartPoint
                   .Y);
30         }
```

```
31            GL.End();
32        }
33    }
```

3.6.5 Zeichnen der dB-Punkte

Für die Darstellung der einzelnen dB Regionen, ergab sich die Wahl eines Punktegitters. Die im Punktegitter gespeicherten Punkte mit dB-Wert werden mithilfe der *DrawNoiseGrid* Funktion gezeichnet. Dabei wird mittels eines *foreach*-Loops über alle gespeicherten Punkte auf dem Gitter gegangen. Den Punkten wird eine Farbe zugewiesen, welche vom Lärmpegel des Knotens abhängig ist. Der Punkt wird schließlich an der gespeicherten Position mit der ermittleten Farbe gezeichnet.

Zeichnen des Punktegiiters

```
1  public void DrawNoiseGrid()
2  {
3      GL.PointSize(10);
4      GL.Begin(PrimitiveType.Points);
5      foreach (NoisePoint noisePoint in NoiseSimulation.NoisePoints)
6      {
7          GL.Color3(GetNoiseArea(noisePoint.NoiseLevel));
8          GL.Vertex2(noisePoint.Location.X, noisePoint.Location.Y);
9      }
10     GL.End();
11 }
```

Die Farbzuweisung inszeniert sich durch die Funktionen *GetNoiseArea* und *GetColor*. Dabei wird zunächst *GetNoiseArea* aufgerufen. An die Funktion werden die dB-Werte, als *double* Typ übergeben. Mithilfe eines *Enums PointColor* werden die unterschiedliche dB-Werte in verschieden Farbbereiche eingegliedert.

Zeichnen der Gebäude und des Geländes

```
1   public enum PointColor
2   {
3       dbArea_1 = 75, dbArea_2 = 70, dbArea_3 = 65, dbArea_4 = 60, dbArea_5
            = 55, dbArea_6 = 50
4   }
```

In *GetNoiseArea* wird zunächst überprüft, in welchem dB-Bereich der Punkt befindet. Anhand der Region kann die entsprechende Farbe übergeben werden. Bei einem dB-Wert von 58

wäre das zum Beispiel *PointColor.dbArea_5*. In der *GetColor* Funktion wird dem Bereich eine Farbe zugewiesen. In dem oben genannten Beispiel wäre das die Farbe Gelb. So kann jedem Punkt die entsprechende Farbe zugewiesen werden. Es wurde ebenfalls die Möglichkeit eingebaut einen einzelnen Punkt anzuklicken, um weitere Informationen über diesen zu erhalten. Mithilfe der Methode *RenderCurrentNoise* wird an der Position des Punktes ein kleines Fenster gezeichnet, welches unter anderem den genauen dB-Wert des Punktes und die kürzeste Distanz zur Trasse anzeigt.

3.6.6 Zeichnen von weiteren Elementen

Für alle weiteren Elemente, wie beispielsweise einer Skala, wurde die Klasse NoiseFrame kreiert. Diese beinhaltet einzelne Funktionen zum Zeichnen von Bildrahmen, der Skala, Text und Hintergrund. Die detaillierte Beschreibung dieser einzelnen Methoden würde den Rahmen der Arbeit überschreiten. Deswegen wird auf die einzelnen Methoden nur kurz eingegangen. Die *DrawSkala* Funktion ist für die Skala zuständig, damit der Benutzer einen Überblick über die einzelnen dB-Bereiche bekommt. Bei der Gestaltung der dB-Schritte der Skala wurde sich an den Lärmkarten (Bayrisches Landesamt für Umwelt, 2019) orientiert. Am Ende entsteht folgendes Bild:

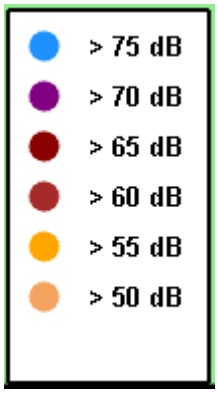


Abbildung 3.4: Lärmskala

DrawMainWindow und *DrawBackground* sind für die allgemeine Gestaltung des Fensters zuständig. Die Funktionen *DrawLeftFrame* und *DrawRightFrame* beinhalten spezifische Gestaltungsoptionen für das jeweilige Fenster, wie zum Beispiel Text. Dieser wird in einer separate Klasse erstellt und an *GL* übergeben.

3.7 Steuerungselemente

Um eine Lärmberechnung durchzuführen, bedarf es an verschiedenen Parametern. Auf dem CDP-Tisch bestand bereits eine Klasse *UIFrame*, womit sich Buttons, Schieber und Text in

ein Steuerungselement implementieren lassen. Für die Eingabe der Parameter wurde entsprechend dieser Klasse ein *UIFrame* entwickelt. Mit einem Schieber lassen sich dort Werte wie DTV eingeben. Es kann ebenfalls festgelegt werden, ob es sich um eine Tag oder Nachtsimulation handelt und ob die Gebäude mit in die Berechnung einfließen sollen. Die programmiertechnische Erstellung des Steuerungselementes ergab sich durch die Erstellung von *UIButton*, *UISlider* und *UIText* Objekten.

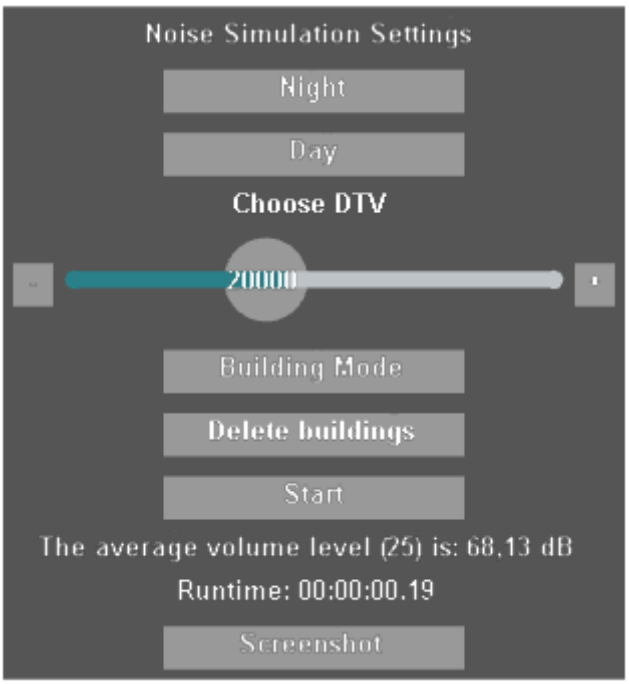


Abbildung 3.5: UI-Frame

3.8 Export mit IFC -Engine

Nach der Platzierung von möglichen Lärmschutzwänden ist es nicht nur notwendig diese grafisch anzuzeigen, sondern auch deren Position zu speichern. Für die Weiterverwendung des Modells bietet sich die Speicherung in ein offenes Datenformat an. Wie unter Abschnitt 2.3.3 geschildert, gibt es kein IFC Objekt spezifisch für Lärmschutzwände, weswegen der Typ IfcWall gewählt wurde. Der Prozess gliedert sich folgendermaßen: Zunächst müssen die Informationen aus dem Modell genommen werden, danach können diese umgewandelt und in das IFC-Dokument übergeben werden. Die Daten der Lärmschutzwand sind in der Klasse *NoiseBarrier* gespeichert. Darin befinden sich sowohl die geometrischen Abmessungen, als auch die Position der Wand.

Klasse *NoiseBarrier*

```
1  public class NoiseBarrier : NoiseBuilding
2  {
3      public Point Location { get; set; }
4      public float Width { get; set; }
5      public float Length { get; set; }
6      public double DistanceToCurve { get; set; }
7      public PointF Direction { get; set; }
8
9      public NoiseBarrier(Point location, float height)
10     {
11         this.Location = location;
12         this.Height = height;
13     }
14     public NoiseBarrier(List<PointF> edgePoints, float height, float
           translationX, float translationY)
15     {
16         this.EdgePoints = TranslateEdgePoints(edgePoints, translationX,
               translationY);
17         this.Height = height;
18         tgis.Edges = CreateDirectionEdges(EdgePoints);
19     }
20     private List<PointF> TranslateEdgePoints(List<PointF> edgePoints,
           float translationX, float translationY)
21     {
22         List<PointF> points = new List<PointF>();
23         foreach (PointF p in edgePoints)
24         {
25             PointF translatedP = new PointF(p.X - translationX, p.Y -
                   translationY);
26             points.Add(translatedP);
27         }
28
29         return points;
30     }
31     private List<PointF> TranslateEdgePoints(List<PointF> edgePoints,
           PointF translationVector)
32     {
33         return TranslateEdgePoints(edgePoints, translationVector.X,
               translationVector.Y);
34     }
35         ...
36 }
```

Um eine IFC- Datei zu verwenden, wird eine zusätzlichen C# Bibliothek benötigt. Dafür wurde die IFC -Engine verwendet. Um die einzelnen Funktionen davon zu verwenden, müssen diese zunächst importiert werden. Die Methoden sind in der dll-Datei IfcEnfine zu finden. Mithilfe des Schlüsselwortes *extern* kann auf die einzelnen Funktionen zugegriffen werden (RDF Ltd., 2019).

Somit ergibt sich das Einlesen wie folgt :

Einlesen externer IFC- Engine Funktionen

```
1  public const string IFCEngineDLL = @"IFCEngine.dll";
2
3  [DllImport(IFCEngineDLL, EntryPoint = "sdaiCreateModelBN")]
4  public static extern Int64 sdaiCreateModelBN(Int64 repository, string
       fileName, string schemaName);
5
6  [DllImport(IFCEngineDLL, EntryPoint = "sdaiSaveModelBN")]
7  public static extern void sdaiSaveModelBN(Int64 model, string fileName);
8
9  [DllImport(IFCEngineDLL, EntryPoint = "sdaiCloseModel")]
10 public static extern void sdaiCloseModel(Int64 model);
11
12 [DllImport(IFCEngineDLL, EntryPoint = "sdaiCreateInstanceBN")]
13 public static extern Int64 sdaiCreateInstanceBN(Int64 model, string
       entityName);
14
15 [DllImport(IFCEngineDLL, EntryPoint = "sdaiPutAttrBN")]
16 public static extern void sdaiPutAttrBN(Int64 instance, byte[]
       attributeName, Int64 valueType,
17     ref double value);
18 ...
```

Für die Speicherung eines Gebäudes ist zunächst eine Hauptmethode *CreateIfcWallStandardCase* erforderlich, die ein IFC-Model einlesen kann und als zweiten input-Parameter ein NoiseBarrier-Objekt besitzt. Die Funktion erstellt zunächst einmal eine IFCWallStandardCase Instanz. Mittels der Methode *PutAttr*, werden dem IFC-Objekt die einzelnen Attribute hinzugefügt. Darunter fallen neben Name und Beschreibung auch, wie unter Kapitel 2.3.3 beschrieben die Platzierung entlang der Achse und die Repräsentation. Anschließend wird das fertige Objekt an die Hauptfunktion als IFC-Handle zurückgegeben. IFC-Handle entspricht einem *Int64*, wurde aber zur besseren Übersicht, entsprechend RDF Ltd. (2019) als IFC-Handle definiert. Die Methode *CreateIfcWallStandardCase* ergibt sich somit wie folgt:

Methode zu Erstellung einer IfcWallStandardCase

```
1  private IfcHandle CreateIfcWallStandardCase(IfcHandle model, IfcHandle
       curve, NoiseBarrier wall)
2  {
3      var ifcWallstandardCase = sdaiCreateInstanceBN(model, "
           IFCWALLSTANDARDCASE");
4      PutAttr(ifcWallstandardCase, "Name", "Laermschutzwand");
5      PutAttr(ifcWallstandardCase, "Description", "Beschreibung: ...");
6      PutAttr(ifcWallstandardCase, "ObjectPlacement", CreateLinearPlacement
           (model, curve, wall.DistanceToCurve, wall.Direction));
7      PutAttr(ifcWallstandardCase, "Representation",
           CreateProductDefinitionShape(model, wall));
8
9      return ifcWallstandardCase;
10 }
```

Um die Attribute der IfcWall hinzuzufügen, müssen zunächst einzelne Instanzen definiert werden. Dafür wurden die Methoden *CreateLinearPlacement* und *CreateCreateProductDefinitionShape* generiert. Für die Erstellung eines IfcLinearPlacement bedarf es ebenfalls einer Hauptinstanz, welche mittels einer IFC-Engine Funktion erstellt wurde. Anschließend werden dem Objekt die einzelnen Attribute hinzugefügt Dazu wurden die Funktionen *CreateIfcDistanceExpression*, *CreateIfcOrrientationExpression* und *CreateIfcAxis2Placement* definiert.

So können alle Attribute des IfcLinearPlacements beschrieben werden.

Methode zur Erstellung eines IfcLinearPlacements

```
1  private IfcHandle CreateLinearPlacement(IfcHandle model, IfcHandle curve,
       double distance, PointF direction)
2  {
3      var ifcLinearPlacement = sdaiCreateInstanceBN(model, "
           IFCLINEARPLACEMENT");
4      PutAttr(ifcLinearPlacement, "PlacementRelTo", curve);
5      PutAttr(ifcLinearPlacement, "Distance", CreateIfcDistanceExpression(
           model, distance));
6      PutAttr(ifcLinearPlacement, "Orientation",
           CreateIfcOrientationExpression(model, direction));
7      PutAttr(ifcLinearPlacement, "CartesianPosition",
           CreateIfcAxis2Placement(model));
8
9      return ifcLinearPlacement;
10 }
```

Die genaue Erläuterung aller Methoden würde über den Rahmen dieser Arbeit hinausgehen. Kurz gesagt sind die Funktionen für die Erstellung der einzelnen IFC-Instanzen zuständig. Innerhalb dieser Methoden werden auch weitere Funktionen wie beispielsweise *CreateIfcDirection* aufgerufen um alle Attribute zu befüllen. Somit ergibt sich ein Verschachtellungsprinzip, worin jede Funktion für eine Sache zuständig ist. Um eine Vorstellung über den Aufbau zu bekommen folgt eine Beschreibung eines Beispiels.

Ein Attribut von *LinearPlacement* ist IfcOrientationExpression. Dieses wird durch die Funktion *CreateIfcOrientationExpression* beschrieben. In der Methode wird eine Instanz IfcOrientationExpression erstellt und die Attribute für die Horizontale und Vertikale definiert. Dafür wird die Funktion *CreateIfcDirection* benötigt, welche ein IfcDirection-Objekt erstellt. IfcDirection wird mittels IfcReal Elementen definert, wofür die Methode *CreateIfcReal* definert wurde. So hat jede Funktion ihr eigene Zuständigkeit und es wird vermieden, dass einzelnen Prinzipien doppelt definiert werden. Die vollständige Struktur des beschriebenen Beispiels wird im folgenden Code Ausschnitt verdeutlicht:

Einzelen Methoden für die Erstellung von IfcOrientationExpression

```
1  private IfcHandle CreateIfcOrientationExpression(IfcHandle model)
2  {
3      var ifcOrientationExpression = sdaiCreateInstanceBN(model, "
           IFCORIENTATIONEXPRESSION");
4      PutAttr(ifcOrientationExpression, "LateralAxisDirection",
           CreateIfcDirection(model, Axis.X));
5      PutAttr(ifcOrientationExpression, "VerticalAxisDirection",
           CreateIfcDirection(model, Axis.Y));
6
7      return ifcOrientationExpression;
8  }
9
10 private IfcHandle CreateIfcDirection(IfcHandle model, Axis axis)
11 {
12     var ifcDirection = sdaiCreateInstanceBN(model, "IFCDIRECTION");
13     CreateIfcReal(ifcDirection, "DirectionRatios", axis);
14     return ifcDirection;
15 }
16
17 private void CreateIfcReal(IfcHandle instance, string title, Axis axis)
18 {
19     var ifcReal = sdaiCreateAggrBN(instance, title);
20
21     switch (axis)
22     {
```

```
23              case Axis.X:
24                  Append(ifcReal, 1);
25                  Append(ifcReal, 0);
26                  Append(ifcReal, 0);
27              break;
28              case Axis.Y:
29                  Append(ifcReal, 0);
30                  Append(ifcReal, 1);
31                  Append(ifcReal, 0);
32              break;
33              case Axis.Z:
34                  Append(ifcReal, 0);
35                  Append(ifcReal, 0);
36                  Append(ifcReal, 1);
37              break;
38          }
39      }
```

Neben der richtige Platzierung der IfcWall, welche durch IfcLinearPlacment erfolgt, muss diese auch geometrisch repräsentiert werden. Da es sich bei den verwendeten Blöcken, die auf den CDP-Tisch gesetzt werden und Lärmschutzwände beschreiben, um keine Objekte mit sonderlich komplexer Geometrie handelt, können diese mit IfcBoundingBox beschrieben werden. Im Laufe der Weiterarbeit mit dem IFC-Model kann diese dann durch komplexere Geometrie ersetzt werden. Zunächst einmal wird die Geometrie mittels der Funktion *CreateProductDefinitionShape* definiert. Darin wird eine Instanz IfcProduktDefinitionShape erstellt. Anschließend wird diesem unter anderem das Attribut zur geometrischen Repräsentation mittels *PutAttr* und *CreateIfcBoundingBox* hinzugefügt. *CreateIfcBoundingBox* erstellt nach dem oben genannten Prinzip eine Instanz von IfcBoundingBox, wofür wieder weitere Funktionen verwendet werden. Die Beschreibung der Geometrie ergibt sich somit wie folgt:

Methode zur Erstellung von IfcProductDefinitionShape

```
1  private IfcHandle CreateProductDefinitionShape(IfcHandle model,
       NoiseBarrier wall)
2  {
3      var ifcWallRepresentation = sdaiCreateInstanceBN(model, "
           IFCPRODUCTDEFINITIONSHAPE");
4      PutAttr(ifcWallRepresentation, "Name", "Geometrische Repraesentation"
           );
5      PutAttr(ifcWallRepresentation, "Description", "...");
6      PutAttr(ifcWallRepresentation, "Representations",
           CreateIfcBoundingBox(model, wall));
```

```
7
8       return ifcWallRepresentation;
9   }
```

Schlussendlich ergibt sich ein Objekt der IfcWall, welches als Lärmschutzwand, inklusive Name, Beschreibung, linearer Platzierung und geometrischer Beschreibung definiert wird. Dadurch kann eine Weiterarbeit mittels unterschiedlicher BIM-Software gewährleistet werden, wodurch sich unbegrenzte Möglichkeiten der Modifizierung ergeben.

Kapitel 4

Beispiele

4.1 Ergebnisse

Mithilfe des erstellten Tools kann nun eine Lärmberechnung durchgeführt werden. Dazu muss zunächst eine Region mithilfe OpenStreetMaps und des CDP-Tisches ausgewählt werden. Diese wird dann an das Noise-Plugin übergeben. Mithilfe des InteractiveAlignment Plugins kann gleichzeitig eine Trasse gezeichnet werden. Darauf folgt die Eingabe der Startparameter. Anschließend kann eine neue Simulation gestartet werden.

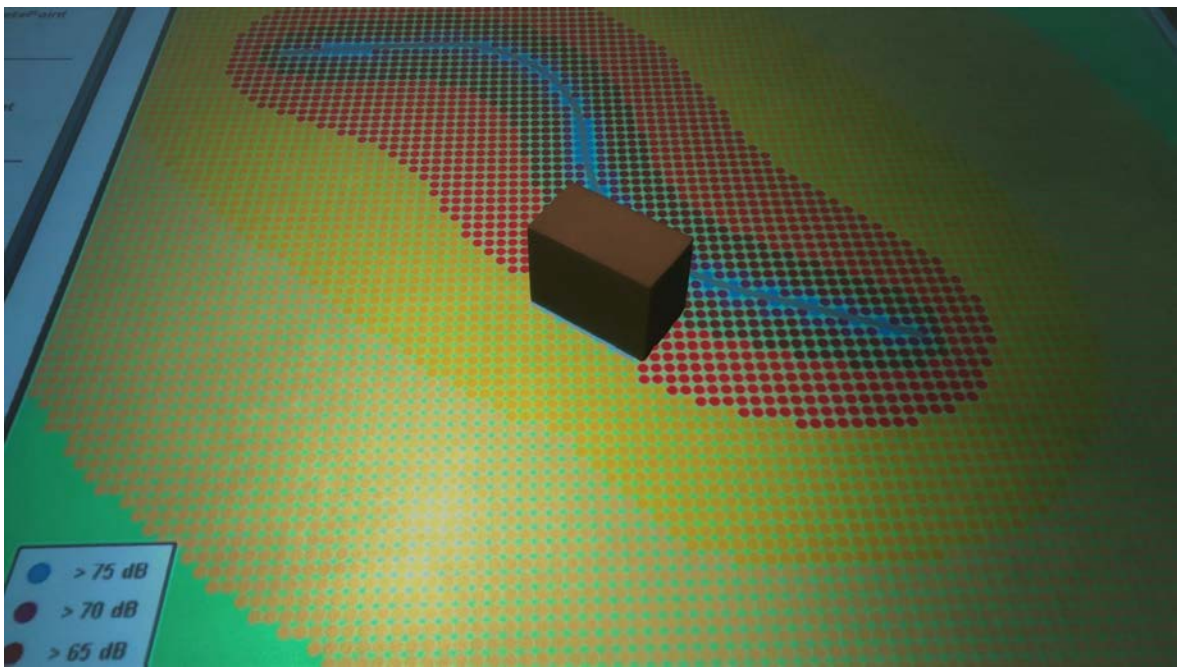


Abbildung 4.1: Lärmsimulation am CDP-Tisch

Nun erfolgt die Bestimmung einer geeigneten Position für eine Lärmschutzeinrichtung. Wenn dort eine Wahl getroffen wurde, wird eine erneute Lärmberechnung gestartet. Ist der Nutzer mit dem erzielten Ergebnis zufrieden, können sowohl die Lage der Trasse, als auch die Position einer Lärmschutzwand in einer IFC-Datei gespeichert werden. Diese Datei kann schlussendlich für die Weiterverarbeitung in anderer BIM-Software verwendet werden.

4.2 Verleich zu der RLS-90 Norm

Für die Validierung der Ergebnisse ist es wichtig die verwendete Norm genauer zu betrachten. Deswegen folgt ein Vergleich mit einem Beispiel der RLS-90 Norm. Dabei sind die folgenden Eingabeparameter zu berücksichtigen:

- DTV: 20000
- zulässige Höchstgeschwindigkeit: 100 km/h
- Korrektur zur Straßenoberfläche: 0 dB
- LKW-Anteil (tags): 10%

Der Immissionspunkt ist dabei 145,4 Meter vom Fahrstreifen und somit dem Emissionsort entfernt. Nach der Beispielrechnung der RLS-90 Norm ergeben sich die folgenden Werte:

- Emissionspegel Fahrstreifen (tags): 67,7 dB
- Pegeländerung durch unterschiedliche Abstände: 7,1 dB
- Beurteilungspegel (ohne Gebäude): 60,6 dB

Mithilfe des Lärmtools wird nun versucht die beschriebene entprechende Situation darzustellen. Dafür wurde mithilfe des Interactive Alignment Tools zunächst eine Trasse gezeichnet und anschließend eine Lärmsimulation durchgeführt. Schließlich ergibt sich folgendes Bild:

Abbildung 4.2: Beispielrechnung

Um die Werte besser überprüfen zu können, wurde die Möglichkeit eingebaut, einen bestimmten Punkt auszuwählen, um dessen Werte anzeigen zu lassen. Entsprechend des Immissionsortes der RLS-90 Norm wurde ein Messpunkt in ca. 145 Meter Abstand ausgewählt. Der Beurteilungspegel an der Trasse liegt bei 60,82 dB(A), die Pegeländerung durch unterschiedliche Abstände hat ein Höhe von 7,08 dB(A) und der Emissionspegel liegt bei 67,90 dB(A). Die geringen Abweichungen zu den Werten der RLS-90 Beispielrechnung ist damit zu begründen, dass es schwierig ist einen Punkt mit exakt gleichem Abstand auszuwählen. Um die Genauigkeit und Korrektheit des Plugins weiter zu beurteilen wird im folgenden Abschnitt ein weiteres Beispiel der Lärmberechnung, anhand eines Lärmaktionsplans, beschrieben und mithilfe des Lärmtools simuliert.

4.3 Vergleich zu den Lärmkarten

4.3.1 Beschreibung des Beispiels

Aufgrund zu hoher Lärmbelastungen wurde in der Gemeinde Feldkirchen-Westernham ein Lärmaktionsplan im Jahr 2016 aufgestellt. Die Region liegt im Westen des ostbayrischen Landkreises Rosenheim. Die Gemeinde setzt sich aus 54 Ortsteilen zusammen. Die Hauptverkehrsader bildet die Staatsstraße ST2078, welche durch die Gemeinde entlang des Mangfalltals von Nordwesten nach Südosten verläuft. Dabei kommt es zu einem jährlichen Kraftfahrzeug - Aufkommen von mehr als 3 Millionen Fahrzeugen. Entsprechend der EU-Umgebungsrichtlinie werden im Lärmaktionsplan lediglich die überregionalen Straßen in diesem Gebiet, somit nur die St2078 berücksichtigt (C. Hentschel Consult Ingenieurgesellschaft mbH, 2016).

Diese ist lärmtechnisch mit folgenden Datensätzen beschrieben:

- DTV : 10000 KFZ/a
- Korrektur zur Straßenoberfläche: 0 dB
- zulässige Höchstgeschwindigkeit: 100 km/h
- LKW-Anteil: 10%

Insgesamt sind in der Gemeinde Feldkirchen 350 Personen, was 3,5 % der Region entspricht, von Umgebungslärm betroffen. Um ein genaueres Bild von der Situation zu bekommen wurde der östliche Ortseingang der Stadt Feldkirchen zur Betrachtung gewählt, welcher ebenfalls an der Staatsstraße St2078 liegt.

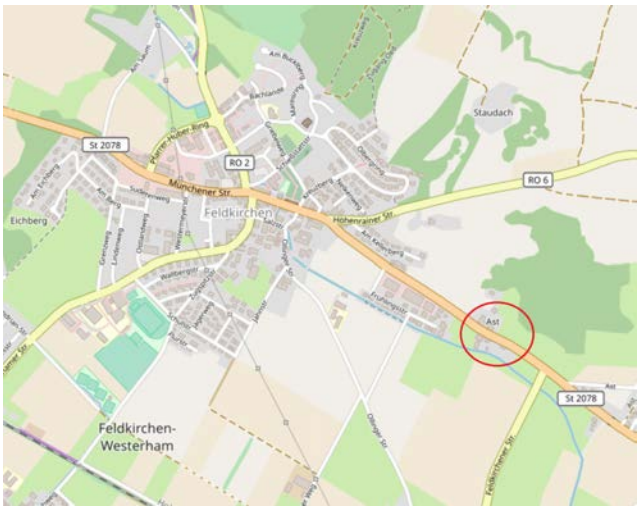


Abbildung 4.3: Lage östlicher Ortseingang (Fossgis e.V., 2019)

Dort sind 8 Wohngebäude mit insgesamt 30 Personen von Umgebungslärm betroffen. Entsprechend der bayrischen Lärmkarten sieht die Lärmlage nachts (22-6 Uhr) wie folgt aus:

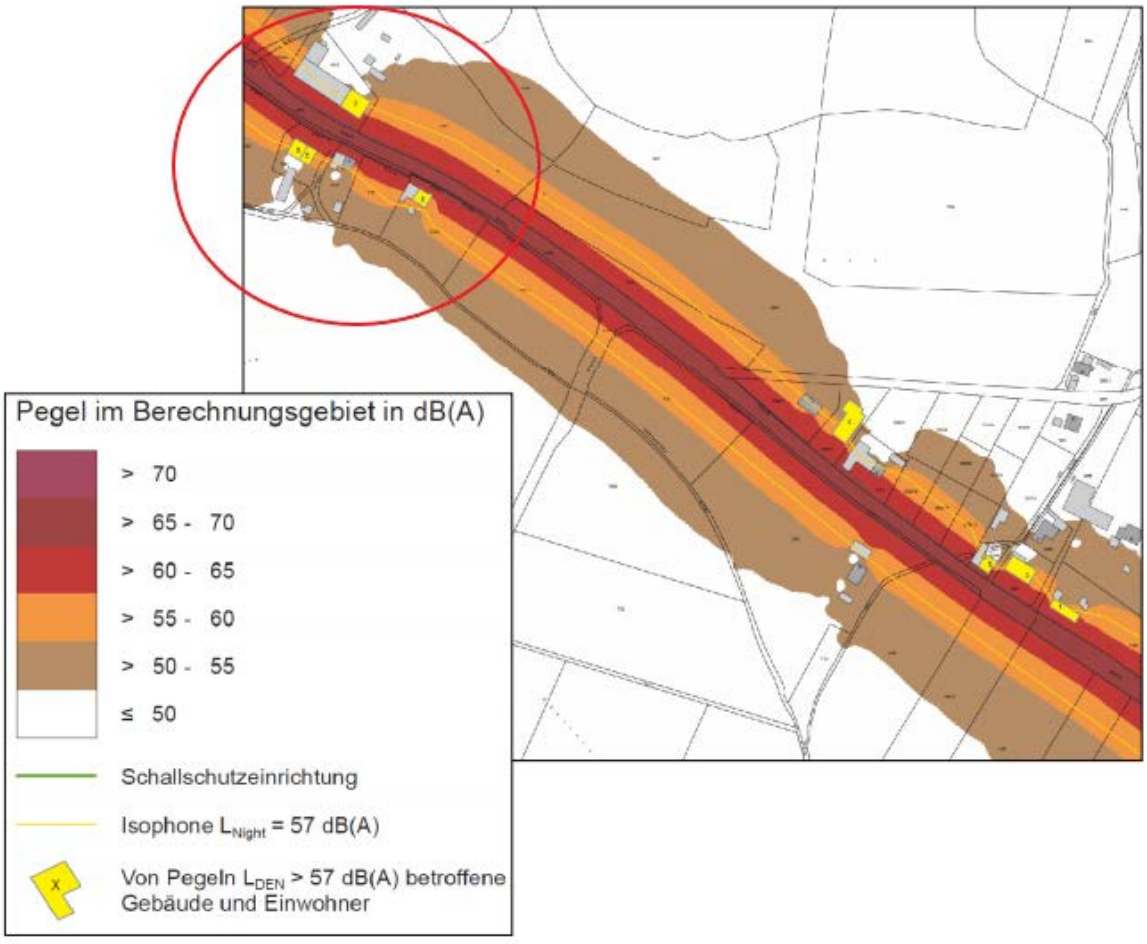


Abbildung 4.4: Lärmausbreitung der Stadt Feldkirchen am östlichen Ausgang (C. Hentschel Consult Ingenieurgesellschaft mbH, 2016)

Daran ist erkennbar, dass die Lärmausbreitung das Gebiet stark belastet. Das zieht nicht nur gesundheitliche Folgen für die Bevölkerung mit sich, sondern auch eine Minderung der Lebensqualität. In den betroffenen Regionen werden Gehsteige gemieden, Gaststätten und Geschäfte leiden darunter und die Ruhe auf Friedhöfen und Kirchen wird erheblich gestört (C. Hentschel Consult Ingenieurgesellschaft mbH, 2016).

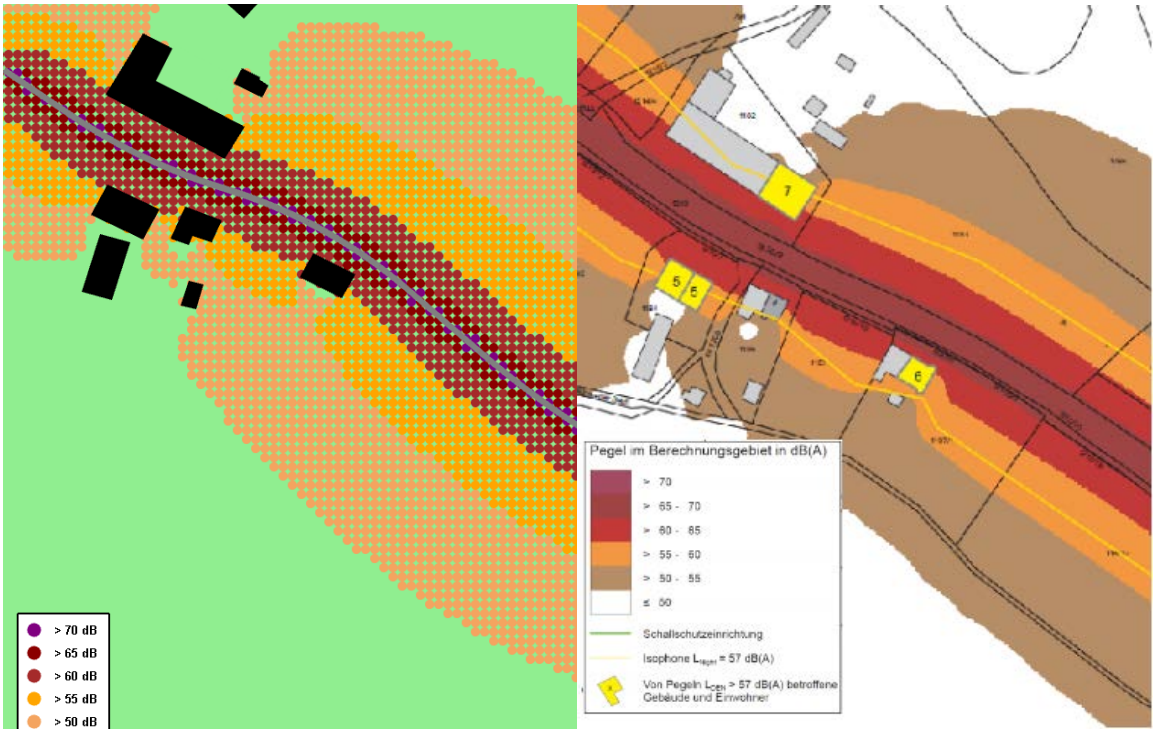
Deshalb hat sich die Gemeinde entschlossen, zukünftige Maßnahmen zur Lärmminderung zu entwerfen. Die Höchstgeschwindigkeit vor dem Ortseingang wurde bereits auf 80 km/h begrenzt. Neben passiven Schallschutzmaßnahmen, wie dem Einbau von schallgedämmten Fenstern, bestehen auch Überlegungen zu aktiven Maßnahmen. Zunächst einmal werden Optionen wie lärmmindernder Asphalt und eine Senkung der Höchstgeschwindigkeit diskutiert.

Sollten diese Maßnahmen nicht ausreichen ist unter anderem auch die Möglichkeit eines Baus von Schallschutzwänden in Betracht zu ziehen. Des Weiteren soll die Lärmsituation auch langfristig betrachtet werden und somit eine eventuelle Planung einer Umgehungsstraße diskutiert werden (C. Hentschel Consult Ingenieurgesellschaft mbH, 2016).

Für diese Fälle wird im Folgenden eine genauere Untersuchung mithilfe des entwickelten Tools durchgeführt. Doch zunächst folgt ein Vergleich der ermittelten Ergebnisse.

4.3.2 Validierung der Ergebnisse

Als Erstes ist ein Begutachtung der berechneten Daten erforderlich. Dafür werden die unter Kapitel 4.3.1 genannten Daten als Eingabeparameter gewählt. Die Trasse wird mit dem Alignment-Tool gezeichnet. Anschließend wird eine Lärmberechnung durchgeführt. Somit ergibt sich folgendes Bild:



(a) Lärmtool **(b)** Lärmkarte (C. Hentschel Consult Ingenieurgesellschaft mbH, 2016)

Abbildung 4.5: Vergleich

Trotz großer Ähnlichkeit zu den Lärmkarten gibt es in einigen Bereichen kleinere Abweichungen. Das liegt zum einen daran, dass die Straßenachse nicht aus den Karten direkt übernommen, sondern entsprechend nachgezeichnet wurde. Des Weiteren wurden die Gebäudedaten aus verschiedenen Quellen hinzugezogen. Allerdings muss in Betracht gezogen werden, dass identische Ergebnisse nicht der Zielsetzung dieser Arbeit entsprächen. Für ei-

ne erste Einschätzung über die Lage der Lärmsituation in einer bestimmten Region ist das erzielte Ergebnis somit als positiv zu bewerten.

4.3.3 Weitere Möglichkeiten des Tools

Einfluss einer baulichen Maßnahme

Mithilfe des Tools kann jetzt über eine mögliche Positionierung der Schallschutzwand entschieden werden. Dafür werden Blöcke verwendet. Ein mögliches Beispiel wäre eine direkte Platzierung entlang der Trasse. Wie unter Kapitel 4.1 beschrieben lässt es sich erkennen, wie sich eine bauliche Schallschutzmaßnahme auf die Region auswirkt. Dabei hat der Benutzer die Option die Wand beliebig zu verschieben und somit einen Überblick über sämtliche Situationen zu erhalten. Diese Flexibilität ist einer der entscheidenden Vorteile des Tools. Ein weiterer Profit des entwickelten Plugins wird im folgenden beschrieben.

Planerische Neugestaltung und lärmschutztechnische Auswirkungen

In kurze Sicht können bauliche Schallmaßnahmen einen enormen Unterschied bezwecken, doch langfristig gestaltet es sich oftmals schwierig. Deshalb empfiehlt die EU-Umgebungsrichtlinie auch andere Schallmaßnahmen, deren Wirkungen sich erst langfristig entfalten wird (EU-Parlament, 2002).

Die Gemeinde ist hauptsächlich von der Hauptverkehrsstraße St2078 betroffen. Anhand eines Flächennutzungsplans werden mögliche Verkehrstrassen zur Entlastung der Orte Feldkirchen und Westernham in Betracht gezogen. Dabei hat sich herausgestellt, dass eine Südumfahrung von Feldkirchen eine deutliche Verkehrsentlastung bewirkt und ebenfalls eine Entlastung der lärmbetroffenen Gegenden erreicht werden könnte (C. Hentschel Consult Ingenieurgesellschaft mbH, 2016).

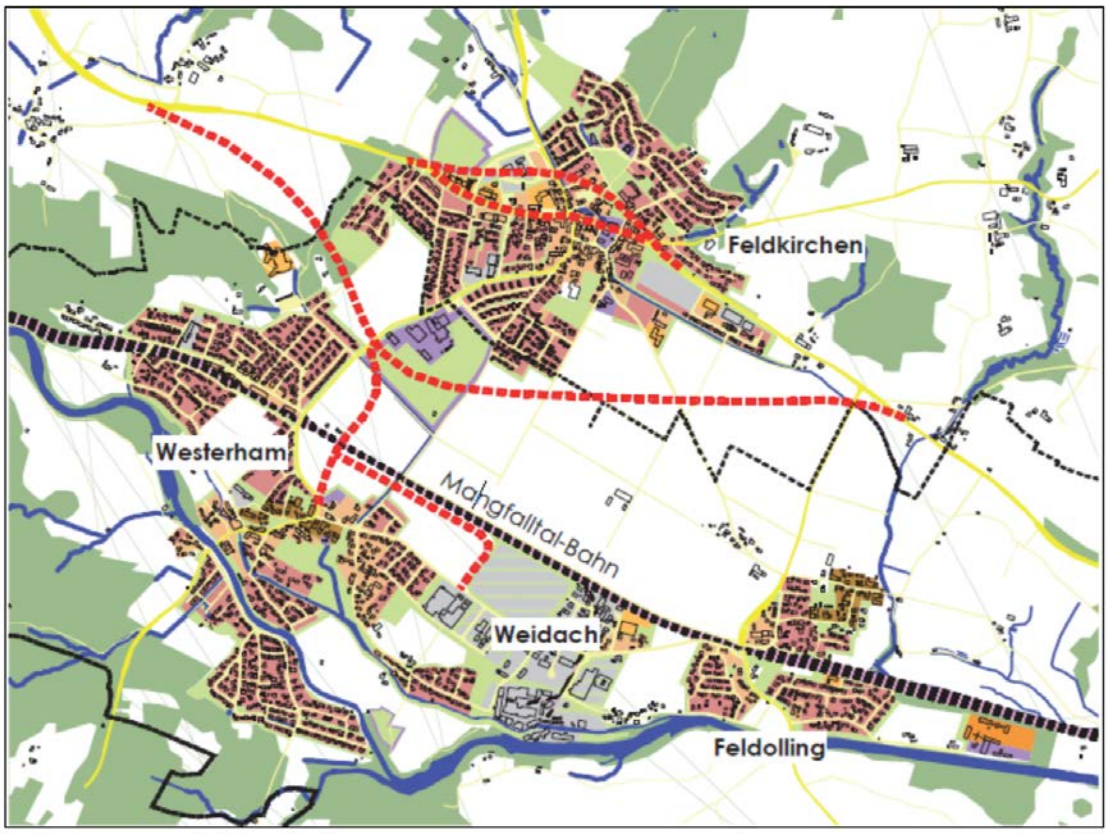


Abbildung 4.6: Lage östlicher Ortseingang (C. Hentschel Consult Ingenieurgesellschaft mbH, 2016)

Für genau so eine Situation ist die Kombination des interaktiven Alignment Tools und des Noise Tools entwickelt worden. Um eine bessere Einschätzung der Lage zu bekommen, wird die Trasse mithilfe des Alignment Tools entsprechend umgeplant. Darauf folgt eine Lärmberechnung. Dabei werden folgende Annahmen für die geplanten Trasse getroffen:

- DTV: 8000 KFZ /a
- Korrektur zur Straßenoberfläche: 0 dB
- zulässige Höchstgeschwindigkeit: 100 km/h
- LKW-Anteil: 9 %

Schließlich ergibt sich folgendes Bild:

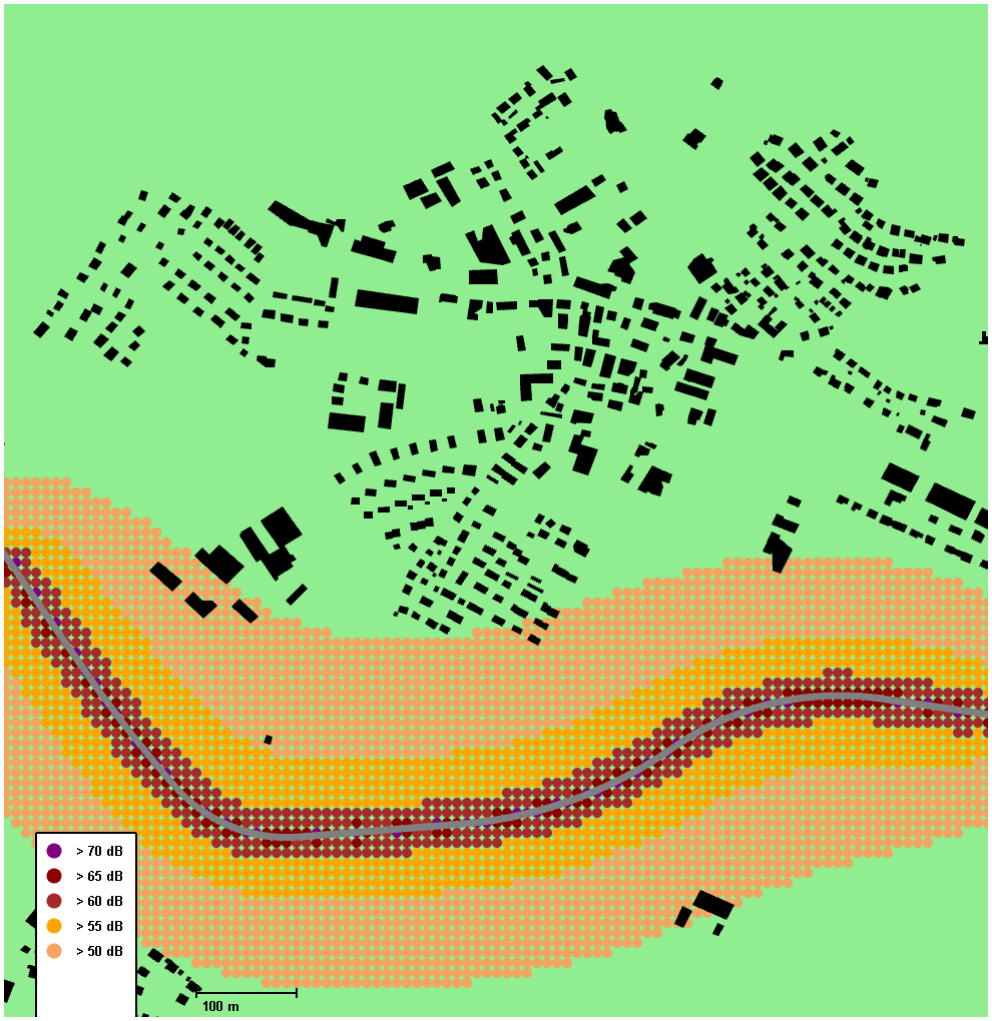


Abbildung 4.7: Mögliche Umfahrung der Ortschaft Feldkirchen

Bei dieser Simulation ist anzumerken, dass die Gebäudedaten nicht mit in Betracht gezogen wurden, um ein schnelleres Ergebnis zu erzielen. Folglich bekommt man schnell einen Überblick, welche Gebiete überhaupt von der Lärmausbreitung betroffen sind. Mit diesen Informationen kann man aus lärmtechnischer Sicht die optimale Lage für eine Trasse bestimmen. Wenn die Position und der Verlauf der Achse bestimmt sind, können diese schließlich in einer IFC -Datei gespeichert werden.

Kapitel 5

Schlussgedanken

5.1 Fazit

Der Kerngedanke dieser Arbeit war es, mithilfe eines Tools den Prozess von Lärmberechnungen zu vereinfachen und optimieren. Mithilfe des Plugins ist dies nun möglich. Veraltete Verfahren können hiermit, wie folgt beschleunigt werden.

Zunächst einmal wird durch das Tool die Möglichkeit geboten ein Lärmbild entlang einer Trasse zu erstellen. Dieses kann durch Eingabeparameter angepasst werden. Durch die Visualisierung bekommt der Nutzer einen schnellen Überblick über die jeweilige Lärmsituation. Ist eine genauere Analyse erforderlich können Daten über die OpenStreetMap Karten miteingezogen werden. Diese fließen in die Berechnungen mit ein, wodurch sich ein detailierteres Bild zeigt. Falls der Bedarf besteht eine Lokalität genauer zu prüfen gibt es die Möglichkeit mittels Touchgesten einen Bereich anzuklicken. Daraufhin werden weitere Informationen visualisiert.

Das Kernziel der Arbeit war es einen ituitiven Entwurf für Lärmschutzwände zu gestalten. Mittels Blöcken und 3D-Objekterkennung wurde dies ermöglicht. Infolgedessen gestaltet sich ein großer Rahmen an Flexibilität der Verwendung des Tools: Blöcke, die Schallschutzmauern symbolisieren, können frei platziert werden. Dadurch bieten sich uneingeschränkte Möglichkeiten der freien Positionierung und Planung. Damit wird nicht nur ein Überblick über die Lärmsitutation an einer Trasse visualisiert, sondern auch der Einfluss einer Schallschutzmaßnahme. Zur Weiterverwendung bieten sich zwei Möglichkeiten. Zunächst kann ein Screenshot erstellt werden, um den Einfluss der Lärmschutzwand und das resultierende Lärmbild als Abzug festzuhalten. Zugleich kann die genaue Position der Schallschutzeinrichtung in einer IFC-Datei gespeichert werden. Durch diese beiden Möglichkeiten bietet sich ein flexibler Weiterarbeitungsprozess. Im Ganzen betrachtet gliedert sich das Tool somit optimal in den BIM-Prozess ein.

Insgesamt wurde somit zur Geltung gebracht, was der entscheidende Vorteil des Tools ist: Die Gestaltung eines schnellen, einfachen Planungsprozesses.

Zusammenfassend ist das in der Arbeit entwickelte Tool auf einem guten Stand, jedoch noch keineswegs perfekt. Es bieten sich noch viele Möglichkeiten zur Erweiterung und Optimierung des Tools. Darauf wird im folgenden Abschnitt eingegangen.

5.2 Ausblick

5.2.1 Lärmsimulationen mit Virtual Reality

Vier wissenschaftliche Mitarbeiter der Chou Universität in Tokyo, Japan haben eine virtuelle Lärmsimulation an einer Bahnlinie implementiert. Dabei werden die Lärminformationen und Störgeräusche in Echtzeit als akustische Signale an die Testperson übertragen. Ziel dabei war es, eine möglichst echte Geräuschempfindung darzustellen. Dafür wurde modernste VR-Technologie verwendet, um neben dem auditiven Eindruck auch eine realistische visuelle Darstellung zu bieten. Die dB-Level wurden mithilfe des genauen Standorts der Person und des Zuges berechnet (Kouji *et al.*, 2018).

Das ganze System besteht aus drei großen Leinwänden, die von Projektoren angestrahlt werden. Es gibt einen Hauptcomputer, der die Kontrolle über vier Teilcomputer hat. Einer der vier kleinen Computer ermittelt die genaue Position der Testperson in Bezug auf die Eisenbahnlinie. Die anderen drei Teilcomputer sind jeweils an die Projektoren angeschlossen und übergeben die Bilder. Das System besteht ebenfalls aus sieben Lautsprechern, die einen intensiven Lärmeindruck erzeugen. Das Bewertungssystem besteht aus einem visuellen und auditiven Teil. Zuerst werden Parameter wie Zugeigenschaften, Immissionspegel und Position des Betrachters bestimmt. In jeder Schleife werden diese Parameter neu berechnet und anschließend simuliert. Für die Visualisierung machten sie Gebrauch von den OpenGL und Cave Bibliotheken. Um die Geräusche virtuell zu simulieren wurden eine stereo Raum-Feld Methode, basierend auf *Ambisionics* verwendet. Dadurch konnte eine komplett ummantelte, reale Geräuschatmosphäre ermöglicht werden (Kouji *et al.*, 2018).

Eisenbahngeräusche klassifizieren sich in zwei Kategorien. Zum einen die Geräusche die durch den Zug, wie beispielsweise Motorgeräusche entstehen, zum anderen den Lärm, der durch den Kontakt zur Schiene bei der Überfahrt des Zuges entsteht. Bei dem Versuch wurden die Daten an der Minami-Furuya Station in Japan ausgewertet. Die Geschwindigkeit des Zuges betrug 83 km/h, gemessen wurde am Gelenk des Gleises. Um eine höhere Genauigkeit zu erzielen, wurden die Messungen mit verschiedenen Wagentypen (motorisiert/nicht motorisiert) ausgeführt. Dabei wurde beobachtet, dass sich bei dem Verknüpfungspunkt von Schiene und

Schwelle die Pegelspitze zeigte. Das wurde mit dem auffallenden Geräusch begründet, welches entsteht, wenn Züge über die Schwellen fahren (Kouji *et al.*, 2018).

Die Lärmberechnungen erfolgten mithilfe einer Punktberechnung in einem freien Feld anhand des ASJ RTN-Model 2008 (Architektur-Norm aus Japan). Ähnlich wie bei der RLS-90 Norm werden dort zunächst die Immissionspegel der Waggons ermittelt und anschließend logarithmisch addiert. Folglich werden die Distanzen zu dem Emissionspunkt bestimmt und daraus ein Bild erstellt. Anhand der Ergebnisse wurde die auditive Berechnung durch C++ Implementierungen mithilfe eines Stereosystems dargestellt. Die visuelle Darstellung wurde durch 3D-Modellierungssoftware generiert. Durch ein Beispiel wurden die Ergebnisse gezeigt. Bei einem Abstand von 9 Metern und einer Geschwindigkeit von 83 km/h ergab sich auf 1,5 Meter Höhe ein maximaler Pegel von ca. 84 dB und Abweichungen zur tatsächlichen Messung von einem Dezibel (Kouji *et al.*, 2018).

5.2.2 Erweiterung des Lärmtools

Simulation an Schienenwegen

Nun stellt sich die Frage, ob es möglich ist, das im Rahmen dieser Arbeit entwickelte Noise-Tool auch für Eisenbahnen zu erweitern. Dafür ist zunächst einmal erforderlich, anhand einer deutschen Norm die Lärmberechnung eines Punktes zu implementieren. Damit keine vollständig neue Implementierung nötig ist, wurde ein C#-Interface *INoiseSimulation* generiert. Dadurch spielt es für alle weiteren Klassen keine Rolle, ob es sich bei der Lärmsimulation um eine straßen- oder schienenbasierte Berechnung handelt. Da viele Ähnlichkeiten bei der Implementierung bestehen, können die bereits erstellten Klassen mehrfach genutzt werden. Die Implementierung sieht dabei folgendermmaßen aus:

Interface: INoiseSimulation

```
1 public interface INoiseSimulation
2 {
3     void Start();
4     void GenerateGridPoints();
5     void GetCurveSegments();
6     void GetSegmentLength();
7     double CalculateBuildings();
8     double CalculateNoiseBarriers();
9 }
```

Das Interface gibt ein Leitbild, wie die zukünftigen Klassen auszusehen haben. Die Methoden der *INoiseSimulation* sind für die vollständige Simulation später notwendig. Eine der Funktionen des Interfaces ist *Start*. Jede Lärmsimulation benötigt eine Startfunktion, um

eine neue Berechnung zu starten. Des Weiteren muss jede Simulationsklasse eine Möglichkeit besitzen, Gitterpunkte zu generieren und die Option bieten, Lärmschutzwände mit einzubeziehen. Dennoch bietet sich die Freiheit, neben der vorgegebenen Methoden und Members noch weitere Funktionen spezifisch für die Simulation zu implementieren. Bei den Berechnungen an der Trasse sind das zum Beispiel die Faktoren LKW-Anteil und durchschnittliches Jahresverkehrsaufkommen.

klassenspezifische Members und Properties

```
1 private readonly double _truckShare;
2 public double AverageTraffic { get; set; }
```

Die Berechnung der Schallausbreitung an der Bahnstrecke erfolgt in Deutschland mit der Schall-03-Norm. Darin gibt es ebenfalls ein ähnliches Berechnungsverfahren, wie an der Trasse. Anhand einer Teilstückzerlegung werden einzelnen Punkte für Schallquellen gebildet. Für die Ermittlung des Beurteilungspegels bilden diese Punktschallquellen das Fundament. Alle linien- und flächenförmigen Quellen werden in einzelne Emissionsquellen zerlegt. Dafür ist es erforderlich einen Immissionsort mit gleichmäßigen Schallausbreitungsbedingungen zu wählen. Um exakte Ergebnis zu erreichen, müssen die Teilstücklängen begrenzt werden. Dabei darf sich der Immissionsanteil für alle Beiträge am jeweiligen Immissionsort um maximal 0,1 dB verändern (Bundesministerium für Verkehr, 2003).

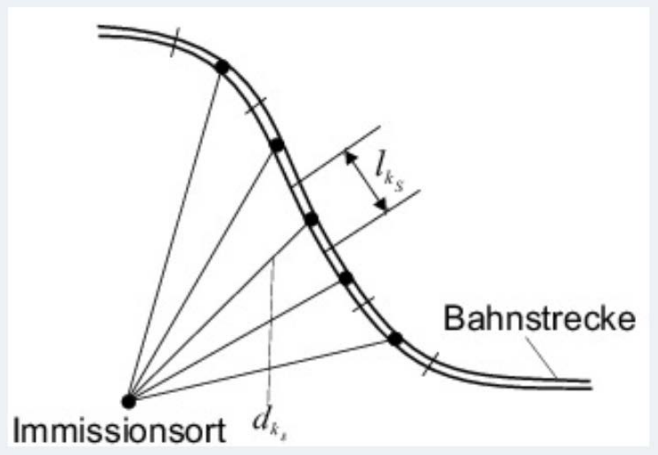


Abbildung 5.1: Teilstückverfahren in der Schall-03-Norm, mit l_{ks} : Länge eines Teilstücks und d_{ks} : Abstand Immissionsort zu Emissionsort (Bundesministerium für Verkehr, 2003)

Miteinbezug eines Geländemodells

Für zukünftige Genauigkeiten ist es natürlich auch von hoher Wichtigkeit, ein Geländemodell miteinzubeziehen. Durch die Verwendung von Interfaces bietet sich nicht nur die Möglichkeit, neue Simulationen zu erstellen, sondern auch bestehende einfach zu erweitern. Um ein

Geländemodell miteinzubeziehen müssen dennoch die Berechnungen angepasst werden. Die Mitaufnahme des Geländes sprengt den Rahmen der RLS-90 Norm. Für diesen Fall muss sich auf eine neue Norm gestützt werden und neue Formeln implementiert werden. Dennoch bleibt die grundsätzliche Gitterberechnung gleich. Folglich müssen die erforderlichen Methoden nur in der Klasse *NoiseFunctions* implementiert werden und an entsprechender Stelle in der *NoiseSimulation* Klasse aufgerufen werden.

Virtual Reality

Die Erweiterung für ein Virtual Reality (VR) Erlebnis ist möglich, aber mit großem Aufwand verbunden. Wie bei Abschnitt 5.2.1 genannt, müssen die Geräusche selbst erzeugt werden. Dafür sind viel tiefgreifendere Berechnungen nötig. Außerdem kann man nicht mehr von einem durchschnittlichen Verkehr ausgehen, sondern müsste die Straße in Echtzeit überprüfen. Des Weiteren ist die Ursprungskonzeption des Tools nicht für eine auditive Ausgabe gedacht. Die Intention dafür war lediglich einen visuelle Überblick über die Ausbreitung von Lärm an einer Trasse zu verschaffen. Da das Plugin für Experten in dem Gebiet gedacht ist, besteht auch nicht die Notwendigkeit, ein auditives Erlebnis zu übermitteln. Insgesamt erscheint eine solche Erweiterung aufgrund des hohen Aufwandes und der fehlenden Brauchbarkeit derzeit als nicht sehr sinnvoll.

Anhang A

C# Programm Code

Der vollständige Code ist unter dem Gitlab des CDP-Projektes zu finden: https://gitlab.lrz.de/cdp/cdp.git

Literaturverzeichnis

Bayerisches Landesamt für Umwelt (2017). *Schall- und Erschütterungsschutz im Planfeststellungsverfahren für Landverkehrswege.*

Bayrisches Landesamt für Umwelt (2018). Straßenverkehrslaerm Bayern. https://www.lfu.bayern.de/laerm/eg_umgebungslaermrichtlinie/kartierung/index.htm. Stand: 20.01.2019.

Bayrisches Landesamt für Umwelt (2019). Lärmbelastungskataster Bayern. http://www.umweltatlas.bayern.de/mapapps/resources/apps/lfu_laerm_ftz/index.html?lang=de. Stand: 20.01.2019.

Borrmann, A., König, M., Koch, C. & Jakob, B. (2015). *Building Information Modelling Technologische Grundlagen und industrielle Praxis.*

buildingSmart International (2019a). IFC-LinearPlacement. http://www.buildingsmart-tech.org/ifc/review/IFC4x1/rc3-20170303/html/schema/ifcgeometricconstraintresource/lexical/ifclinearplacement.htm. Stand: 10.03.2019.

buildingSmart International (2019b). IFC-Road. http://iug.buildingsmart.org/resources/itm-and-iug-meetings-2013-munich/infra-room/ifc-bridge-ifc-for-roads/view. Stand: 10.03.2019.

buildingSmart International (2019). IFC-WALL. http://www.buildingsmart-tech.org/ifc/IFC2x4/rc2/html/schema/ifcsharedbldgelements/lexical/ifcwall.htm. Stand: 20.03.2019.

buildingSmart International (2019). Summary of IFC Releases. =http://www.buildingsmart-tech.org/specifications/ifc-releases. Stand: 02.03.2019.

Bundesministerium für Verkehr (1990). *Richtlinien für den Lärmschutz an Straßen RLS-90.* Forschungsgesellschaft für Strassen-und Verkehrswesen.

Bundesministerium für Verkehr (2003). *Berechnung des Beurteilungspegels für Schienenwege.*

Bundesrepublik Deutschland (1990). Sechzehnte Verordnung zur Durchführung des Bundes-Immissionsschutzgesetzes (Verkehrslärmschutzverordnung–16. BIm-SchV).

C. Hentschel Consult Ingenieurgesellschaft mbH (2016). Laermaktionsplan Feldkirchen Westernham. Stand: 14.02.2019.

EU-Parlament (2002). Richtlinie 2002/49/EG des europäischen Parlaments und des Rates.

Fossgis e.V. (2019). OpenStreetMap. https://www.openstreetmap.de/index.html. Stand: 26.01.2019.

Gemeinschaftsarbeitsausschuss NALS/NATG: Terminologie und Einheiten der Akustik (2009). *DIN 1320, Akustik.*

Häupl, P., Homann, M., Kölzow, C., Riese, O., Maas, A., Höfker, G. & Christian, N. (2017). *Lehrbuch der Bauphysik: Schall-Wärme-Feuchte-Licht-Brand-Klima.* Springer-Verlag.

Ingenieurbüro Grassl (2018). Lärmschutzwand München Freimann. https://www.grassl-ing.de/laermschutzwaende/details/p/show/bab-a9-laermschutzwand-freimann/. Stand: 24.02.2019.

Kouji, K., Toru, Y. & Kazuo Kashiyama, M. S. (2018). *Development of a Railway Noise Evaluation System Using Virtual Reality Technology.*

LaMothe, A. (2002). *Tricks of the Windows game programming gurus.* Sams Publishing.

Liebich, T. & Gersching, E. (2010). IT Unterstützung für die Wertschöpfungskette Bau.

Markič, S., Schlenger, J. & Bratoev, I. (2018). *Tangible Alignment Design.* Technische Universität München, Lehrstuhl CMS.

Microsoft (2019). Microsoft Developer Network.

Mikael, L. & Arto, K. (2012). *The Ifc Standard - A Review Of History, Development, and Standardisation.* Information Systems Science, Hanken School of Economics, Finland.

Ramm, F. & Topf, J. (2010). *OpenStreetMap: Die freie Weltkarte nutzen und mitgestalten.* Lehmanns Media.

RDF Ltd. (2019). IFC-Engine Documentation. http://rdf.bg/ifcdoc/CS64/sdaiOpenModelBN.html. Stand: 15.02.2019.

Schlenger, J. (2018). Interactive Alignment Design.

Schubert, G. (2012). *Early Design Support: Interaktive Simulationen in frühen Entwurfsphasen.* Dissertation, Technische Universität München, Lehrstuhl Architekturinformatik.

Schubert, G. & Petzold, F. (2017). *Visual Programming meets Tangible Interfaces - Generating city simulations for decision support in early design stages.* Technische Universität München, Lehrstuhl Architekturinformatik.

Silicon Graphics (2019). OpenGL API Documentation. http://docs.gl/. Stand: 14.03.2019.

SSF-Ingenieure (2018). Planung und Gestaltung von Lärmschutzwänden.

Abbildungsverzeichnis

Tabellenverzeichnis

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelor-Thesis selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ich versichere außerdem, dass die vorliegende Arbeit noch nicht einem anderen Prüfungsverfahren zugrunde gelegen hat.

München, 12. April 2019

Fabian Pfitzner

Fabian Pfitzner
Dachauerstraße 125
D-80335 München
e-Mail: fabian.pfitzner@tum.de