Hochschule Rosenheim
Fakultät für Informatik

Masterarbeit

Masterprüfung WS 2012/2013

Stefan Zagler

Entwicklung einer 3D-Simulationsumgebung zur Darstellung von Hagelabwehr-Flügen

Erstprüfer: Prof. Dr. Jochen Schmidt
Zweitprüfer: Prof. Dr. Peter Zentgraf

Erklärung

Ich versichere, dass ich diese Arbeit selbständig angefertigt, nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Rosenheim, den 09.01.2013

Stefan Zagler

Kurzfassung

In dieser Masterarbeit wird eine mögliche Implementierung der 3D-Visualisierung von Hagelabwehreinsätzen verwirklicht. Bei Hagelabwehreinsätzen fliegt ein speziell für diesen Einsatz ausgerüstetes Flugzeug in das Zentrum eines Unwetters, bei dem es zu Hagelschauern kommen könnte. Durch das Versprühen von Silberiodid wird versucht, die Intensität des Hagelschauers zu mindern oder diesen in harmlosen Regen umzuwandeln. Die Simulation dient hauptsächlich dazu, die Thematik Hagelabwehr einem größeren Publikum näher zu bringen, aber auch die an der Hagelabwehr Beteiligten sollen davon profitieren. Das Ganze ist ein Teilprojekt von RO-BERTA (ROsenheims meteorologische BEsonderheiten: Eine Regelungs-Technische Aufgabe) unter Leitung von Prof. Dr. Zentgraf, einem Projekt zur Effizienzsteigerung der Hagelabwehrflüge.

Die Simulation wird auf Basis der Grafikengine Unity realisiert. Die Eingangsdaten der Software bestehen aus Informationen zum Gelände, Wetterdaten und den während des Fluges aufgezeichneten Daten. Die Daten zum Gelände wurden vom Autor der Arbeit in Eigenarbeit zusammengesucht, die Wetterdaten stammen vom deutschen Wetterdienst und die Daten aus dem Hagelabwehrflieger werden von der Hochschule Rosenheim zur Verfügung gestellt.

Nach einer entsprechenden Verarbeitung dieser Daten kann dann sowohl ein gerade ein stattfindender als auch ein bereits stattgefundener Hagelabwehreinsatz von der Software grafisch dargestellt werden. Zu sehen ist dabei im Wesentlichen die Landschaft, eine Darstellung der Wetterdaten und das Flugzeug mitsamt einer Repräsentation des bereits zurückgelegten Flugweges.

- Hagelabwehr
- RO-BERTA
- Unity

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Einführung Hagelabwehrflüge

Hagel verursacht jährlich enorme finanzielle Schäden. Um diese zu reduzieren, gibt es in einigen Gegenden Projekte, den Hagel durch den Einsatz von Silberiodid abzuschwächen oder gar ganz zu verhindern. Die Silberiodidpartikel sollen hierbei in den Wolken als weitere Kondensationskeime zusätzlich zu den natürlich vorkommenden dienen. Durch das Hinzufügen von Kondensationskeimen in die Wolke sollte die Menge an Flüssigkeit an einem einzelnen Keim abnehmen und damit auch die Anzahl der großen und schädlichen Hagelkörner. Normalerweise wird das Silberiodid mit Hilfe von Flugzeugen, sogenannten Hagelfliegern, in die Wolken gebracht. So wird es auch im Raum Rosenheim praktiziert. Um die Hagelabwehr weiter voran zu bringen, wurde 1994 der „Verein zur Erforschung der Wirksamkeit der Hagelbekämpfung im Raum Rosenheim e.V“ gegründet (siehe [Hag]). Die Fachhochschule Rosenheim hat zur Steigerung der Effektivität der Hagelabwehreinsätze zusammen mit dem Verein das Projekt „RO-BERTA“ (ROsenheims meteorologische BEsonderheiten: Eine Regelungs-Technische Aufgabe) ins Leben gerufen (siehe [ROB]). Ein kleineres Teilprojekt von RO-BERTA ist die Realisierung einer visuellen Darstellung des Hagelabwehrfluges. Mit genau dieser beschäftigt sich diese Arbeit.

1.2 Projektübersicht

Als Teilprojekt von RO-BERTA ist diese Arbeit auch teilweise abhängig von den anderen Teilbereichen von RO-BERTA. Abb. 1.1 (aus [Ber12b]) zeigt einen Überblick über die verschiedenen Module von RO-BERTA.

Daten werden mit Hilfe von verschiedenen Sensoren während des Fluges aufgezeichnet. Diese werden dann via Digitalfunk an eine extra dafür eingerichte Station am Hochfellngipfel gesendet. Von dort aus werden sie wiederum auf einer WLAN Richtfunkstrecke an Rechner in der FH Rosenheim wei-

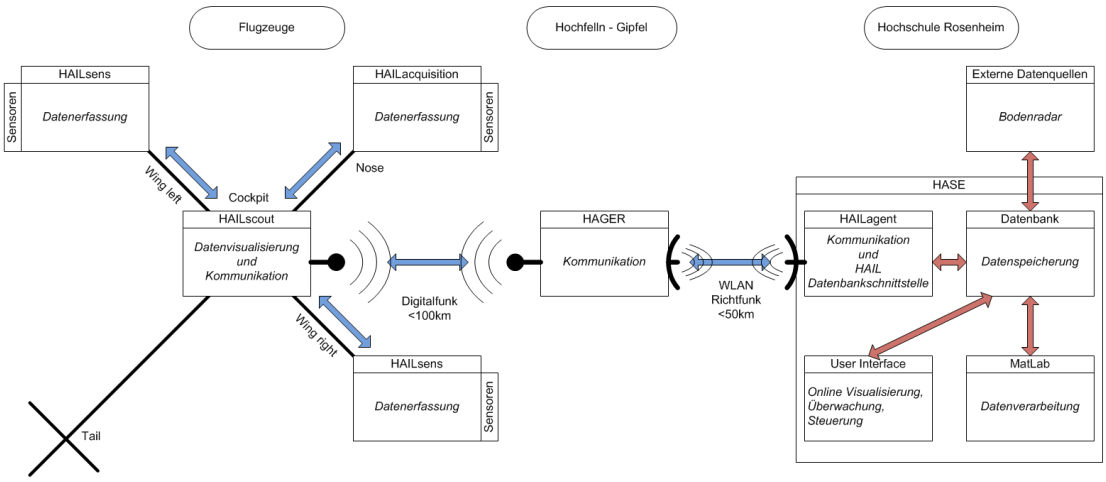


Abbildung 1.1: Gesamtübersicht Module RO-BERTA

tergeleitet. Dort werden sie gespeichert und können dann dementsprechend verarbeitet werden. Das Funktionieren dieser Datenübertragung ist essentiell für Flugvisualisierung, da ohne die Daten vom Flieger auch nichts dargestellt werden kann. Während der Hagelsaison 2012 konnten keine Daten von Hagelabwehrflieger gewonnen werden wie es im laufenden Betrieb vorgesehen ist. Um trotzdem für diese Arbeit verwendbare Daten benutzen zu können, wurden schlussendlich Daten verwendet, die zwar auch während eines Hagelabwehrfluges aufgezeichnet wurden, allerdings eine deutlich niedrigere Datenrate aufweisen.

1.3 Zielsetzung

Die genaue Zielsetzung ist für dieses Projekt von großer Bedeutung. Zunächst einmal ist die Forderung nach einer Simulation eines Hagelabwehrfluges nicht sehr präzise. Für eine erste Eingrenzung des Themas ist die Sondierung der zu erhaltenen Daten am Wichtigsten. Wenn man das Format der Daten kennt und weiß wie man sie bekommt, kann man daraus Rückschlüsse auf die geforderten Funktionalitäten der Software schließen. In diesem speziellen Fall ist auch der Vergleich mit der Diplomarbeit von [Hög09] von Nöten. In dieser wurde bereits im Jahre 2009 eine Visualisierung eines Hagelabwehrfluges bewerkstelligt. Als Grundlage diente damals das Programm Google Earth der Firma Google (siehe [Goo]). Mit diesem Programm ist, im Wesentlichen zusammengefasst, folgendes möglich:

- Das Abspielen eines alten, aufgezeichneten Hagelabwehrfluges mit Zeitsteuerung

- Darstellung der Wettersituation in Form von verschiedenfarbigen Würfeln je nach Wetterintensität
- Visualisierung des Fluges in Form einer Linie entlang des Flugweges
- Anzeige von verschiedenen, während des Fluges aufgenommenen, Wetterdaten

Abbildung 1.2: Flugvisualierung Höglauer

Abb. 1.2 stellt ein Beispiel einer solchen Flugvisualisierung mit dem Programm von Höglauer dar. Darauf zu sehen ist das Gebiet des Hagelabwehreinsatzes, der Flugweg des Hagelabwehrfliegers in Form einer grünen Linie und das Unwetterzentrum über den Zeitraum des Einsatzes, dargestellt durch blaue Würfel. Nun stellt sich natürlich die Frage, warum eine neue Software zur Visualisierung geschrieben werden soll, wenn es doch schon eine gibt. Der wichtigste Punkt dabei ist die simple Tatsache, dass die von Höglauer entwickelte Lösung mit aktuellen Versionen von Google Earth nicht mehr korrekt funktioniert. Zwar kann man mit Hilfe des Programmes erstellte Dateien noch in Google Earth laden, allerdings ist keine Wetterdarstellung mehr vorhanden. Damit ist der Nutzen der Visualisierung natürlich stark eingeschränkt, da die Wetterdarstellung und insbesondere die Information über die Hagelzentren für die Nachbetrachtung von entscheidender Bedeutung sind. Um dies zu verhindern, sollte die in diesen Projekt erarbeitete Lösung möglichst unabhängig von externen Programmen sein.

Ein weiterer wichtiger Gesichtspunkt ist die Tatsache, dass mit der alten Visualisierung keine Darstellung von gerade stattfindenden Hagelabwehreinsätzen möglich ist. Nur bereits absolvierte Einsätze können mit Hilfe der während eines Fluges gesammelten Daten nachgestellt werden. Eine öffentlich zugängliche Livesimulation eines Hagelabwehrfluges ist besonders im Hinblick auf die Öffentlichkeitsarbeit von Interesse. Zum einen könnte dadurch die öffentliche Aufmerksamkeit und somit eventuell auch die finanzielle Förderung gesteigert werden, zum anderen könnte die umstrittene Wirksamkeit von Hagelabwehrflügen besser untermauert werden. Zusammengefasst soll die neue

Realisierung der Visualisierung alle Möglichkeiten der alten Lösung beinhalten, als auch besonders folgende Punkte berücksichtigen:

- Unabhängigkeit des Simulationsprogramms von externen Programmen
- Für alle zugreifbare Livepräsentation eines gerade stattfindenden Hagelabwehrfluges

Wie genau diese Vorgaben zu erfüllen sind, war am Anfang nicht genauer definiert und somit zum Großteil dem Autor dieser Arbeit überlassen. Einzig und allein die Verwendung der Grafikengine Unity als Unterbau für die Simulation war wegen den bereits vorhandenen Erfahrungen von Projektmitgliedern vorgeben. Die Resultate der Arbeit wurden dann jeweils den anderen Mitgliedern des Projektes RO-BERTA vorgestellt und soweit Kritikpunkte oder Verbesserungsvorschläge eingebracht wurden, sind diese soweit möglich berücksichtigt. Mit der genauen Implementierung der Software beschäftigt sich das Kapitel 3.

1.4 Überblick über die Arbeit

Eine kurze Übersicht über die nachfolgenden Kapitel:

Allgemeine Konzepte: Dieses Kapitel beschäftigt sich als erstes mit dem Thema „Abbildungen der Erde“. Erläutert werden dabei die allgemeinen Konzepte dieser Abbildungen. Danach wird konkreter die Gauß-Krüger-Projektion präsentiert. Der zweite Teil des Kapitels gibt einen kurzen Überblick über die verwendete Spiele-Engine Unity.

Implementierung: In diesem Kapitel wird als erstes das Format der vorliegenden Daten diskutiert, sortiert nach den Datenquellen. Im Anschluss daran wird, ebenfalls sortiert nach den einzelnen Datenquellen, genauer auf die grafische Darstellung jener Daten in der Simulation eingegangen.

Ergebnis/Beispiel: In diesem Kapitel wird anhand eines Beispiels die Funktionsweise der Software demonstriert.

Kapitel 2

Allgemeine Konzepte

2.1 Kartennetzentwurf

2-Dimensionale Abbildungen der Erdoberfläche sieht man heutzutage normalerweise in Navigationsgeräten, in Programmen wie GoogleMaps oder auch auf den tradionellen Papierlandkarten. Doch den Wenigsten sind dabei die Besonderheiten dieser Abbildungen bewusst. Denn es ist unmöglich, die Erdoberfläche als 3-Dimensionales Objekt komplett verzerrungsfrei ins 2-Dimensionale abzubilden. Im Lauf der Geschichte wurden etliche Projektionsmethoden für die unterschiedlichsten Einsatzzwecke erstellt. Daher folgt nun ein kurzer Überblick über die verschiedenen Abbildungsarten und dann ein Abschnitt über die Wahl der schlussendlich verwendeten Projektionsart (siehe [Eur]).

2.1.1 Abbildungsarten

Bevor man sich Gedanken über die Art und Weise der Abbildung macht, muss man sich zunächst Gedanken über die Form der Erde machen, also das abzubildende Objekt selbst. Der erste einfache Ansatz wäre dabei die Darstellung der Erde als Kugel. Allerdings entspricht dies nicht der Realität. In Wirklichkeit entspricht die Erdform eher einem abgeplatteten Rotationsellipsoid. Ein solcher ist z.B. im World Geodatic System 1984 (WGS 84) definiert (siehe [WGS]). Dieser Referenzellipsoid weist zwar immer noch Unterschiede zur wahren Erdform auf, ist aber für viele Abbildungen schon eine brauchbares Modell.

Eine mögliche Einteilung der Abbildungen ist die, anhand der gewählten Projektionsfläche. Im wesentlichen sind dies die azimutale, die zylindrische und die konische Projektion. Im Folgenden sollen diese Abbildungsarten näher betrachtet werden.

Abbildung auf Ebene

Bei der azimutalen Abbildung ist die Abbildungsfläche eine 2-dimensionale Ebene. Ein Beispiel dafür ist in Abb. 2.1 dargestellt.

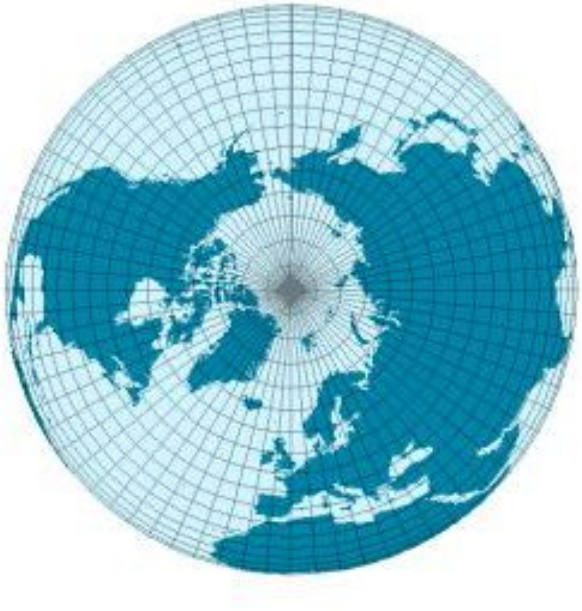


Abbildung 2.1: Abbildung Erde auf Ebene

Alle azimutalen Abbildungen, die sich geometrisch konstruieren lassen, besitzen einen festen Projektionspunkt. Bei der stereografischen Projektion liegt der Projektionspunkt genau gegenüber dem Berührungspunkt von Erde und Abbildungsebene. Ist das Projektionszentrum der Erdmittelpunkt, so spricht man von einer gnomischen Projektion. Bei der orthografischen Projektion liegt das Projektionszentrum im Unendlichen, wodurch die Projektionsstrahlen alle parallel zueinander sind. Es gibt allerdings auch azimutale Projektionen, die rein mathematisch definiert sind und sich nicht geometrisch konstruieren lassen. Ein Beipiel dafür ist die flächentreue Azimutalprojektion.

Abbildung auf Zylinder

Die zylindrische Projektion bildet das Erdmodell auf einen Zylinder ab. Genauer gesagt wird die Erde auf die seitliche Oberfläche des Zylinders projiziert. Das auf dieser Oberfläche abgebildete Areal kann dann ganz einfach auf eine Ebene, d.h. ins 2-dimensionale, übertragen werden. Bei der Übertragung dieses Teiles der Zylinderoberfläche kommt es zu keinen Verzerrungen. Abb.

2.2 zeigt ein Beispiel. Eine weitere Unterteilung der zylindrischen Projektion kann anhand der Lage des Zylinders getroffen werden. Wenn die Zylinderachse die selbe Richtung hat wie die Erdachse, so spricht man von einer normalen Lage. Transversal dagegen bedeutet, dass die Zylinderachse senkrecht zur Erdachse liegt. Alle anderen Lagen werden als schiefachsig bezeichnet.

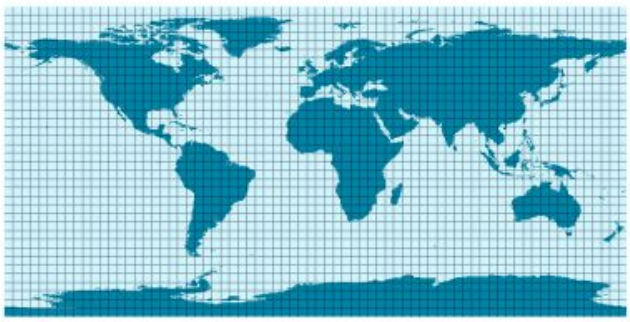


Abbildung 2.2: Abbildung Erde auf Zylinder

Abbildung auf Kegel

Bei der konischen Projektion ist der Hilfskörper, auf dem die Erdoberfläche abgebildet wird, ein Kegel. Wie bei der zylindrischen Projektion wird das Gebiet auf der Oberfläche des Kegels abgebildet und diese Oberfläche schließlich 2-dimensional dargestellt. Eine solche Abbildung ist in Abb. 2.3 zu sehen.

2.1.2 Verzerrungseigenschaften

Wie bereits in der Einleitung erwähnt, haben alle 2-dimensionalen Abbildungen der Erde mehr oder weniger starke Verzerrungen. Abhängig von der gewählten Abbildung gibt es Eigenschaften, die frei von Verzerrungen bleiben. Die drei wichtigsten sollen im Folgenden vorgestellt werden.

Längentreue

Mit Längentreue bezeichnet man die Eigenschaft, dass zwei Vergleichsstrecken, die auf der Erde ein bestimmtes Verhältnis zueinander haben, dieses Verhältnis auch in der Projektion haben. Im konkreten Fall würde dies zum Beispiel bedeuten, wenn Strecke A auf der Erde doppelt so lang wie Stecke B

Abbildung 2.3: Abbildung Erde auf Kegel

ist, dass auch die entsprechende Abbildung von Stecke A und B zueinander im Verhältnis 2:1 stehen. Eine echte Längentreue kann allerdings nur entlang gewisser Linien erreicht werden. Welche Linien dies sind, ist abhängig von der gewählten Abbildungsart. Es kann sich dabei z.B um die Breitenkreise und die Meridiane handeln. Beispielsweise würde bei einer Projektion mit längentreuen Breitenkreisen der oben genannte Vergleich nur dann gelten, wenn die Strecken A und B auf exakt demselben Breitenkreis verlaufen würden. Die Konsequenz daraus ist auch, dass eine Längentreue in alle beliebigen Richtungen bei einer bestimmten Projektion unmöglich ist. Allerdings heißt dies wiederum nicht, dass alle Karten für exakte Längenmessung unbrauchbar sind. Bei Längentreue sind die entsprechenden Verhältnisse zu 100% korrekt. Für Anwendungen in der Realität ist eine solche Genauigkeit kaum erforderlich. Will man Längenmessungen in der Abbildung vornehmen, ist es durchaus hinnehmbar, wenn die Verzerrungen im nicht messbaren oder vernachlässigbaren Bereich liegen. Als grobe Faustregel kann gesagt werden, je größer das abzubildende Gebiet ist, desto größer sind die unvermeidbaren Längenverzerrungen.

Winkeltreue

Ist eine Abbildung winkeltreu, so sind die Winkel zwischen zwei Strecken auf der Erde identisch mit dem Winkel zwischen den entsprechenden projizierten Strecken in der Abbildung. Ein sehr bekanntes Beispiel für eine winkeltreue Abbildung ist die Mercatorprojektion. Diese ist aus der Seenavigation hervorgegangen und ist immer noch eine weitverbreitete Methode, um den gesamten Globus darzustellen (siehe Abb. 2.4):

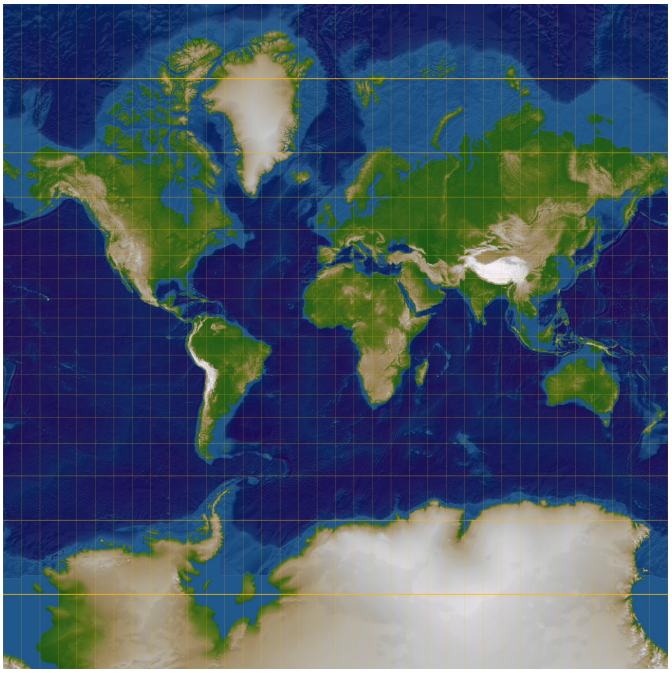


Abbildung 2.4: Mercatorprojektion der Erde

Die Mercatorprojektion ist eine Zylinderprojektion. Die Verzerrungen von Länge und Form sind in der Nähe des Äquators am geringsten und nehmen in Richtung der Pole stark zu. Gebiete nahe der Pole, wie z.B. Grönland, werden überproportional groß dargestellt.

Flächentreue

Eine Abbildung ist genau dann flächentreu, wenn zwei Flächen auf der Erde exakt das selbe Verhältnis zueinander haben wie in der Abbildung. Die Form der Flächen kann dabei mehr oder weniger stark verzerrt sein. Die Verzerrungen sind umso stärker, je weiter die Fläche vom Projektionszentrum entfernt ist.

Vermittelnde Projektion

Mit vermittelnder Projektion ist gemeint, dass die Abbildung so gut wie möglich die Wirklichkeit nachbilden soll. Die Verzerrung von Länge, Fläche und Winkel sollten also in der Summe minimal sein. Diese Art der Projektion erfüllt meist keine der drei weiter oben genannten Treueeigenschaften. Ein Beispiel für die vermittelnde Projektion ist die Winkel-Tripel-Projektion. Abb. 2.5 zeigt die Erde, dargestellt mit Hilfe dieser Projektion.

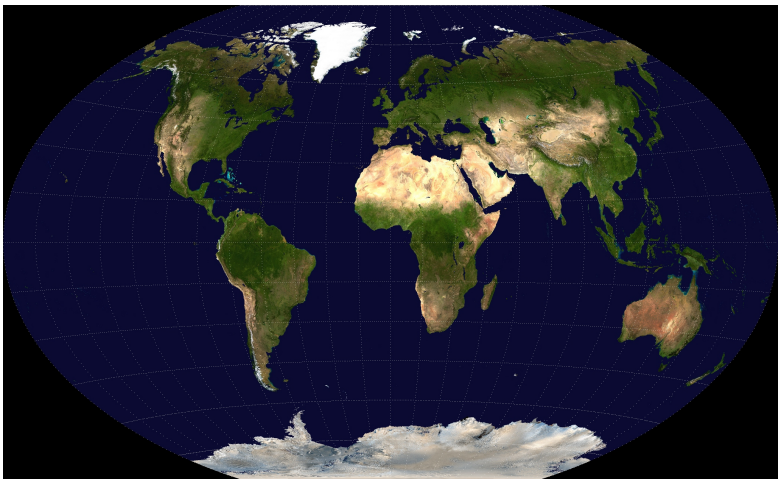


Abbildung 2.5: Winkel-Tripel-Projektion der Erde

2.1.3 Gauß-Krüger-Koordinatensystem

Für die in der Simulation zu verwendenden Daten ist insbesondere eine möglichst gute Längentreue notwendig. Zum einen sollte bei dem benutzten 2-dimensionalen Koordinatensystem auf beiden Koordinatenachsen ein Meter gleich lang abgebildet sein. Zum anderen sollte auch die Verzerrung der Länge auf verschiedenen Gebieten des abgebildeten Geländes minimal sein. Da das darzustellende Terrain nicht allzu groß ist, wird das Gauß-Krüger-Koordinatensystem verwendet. Dieses findet vor allem in Deutschland Verwendung. Die Gauß-Krüger-Projektion ist eine transversale Zylinderprojektion. Die Erde wird also auf einen im Bezug auf die Erdachse quer liegenden Zylinder abgebildet (Abb. 2.6).

Für eine spezifische Gauß-Krüger-Projektion wird immer ein am Anfang definierter Bezugsmeridian verwendet. Die verwendeten Bezugsmeridiane liegen jeweils 3 Längengrade auseinander. Bei allen Gebieten, die in der Nähe dieses Bezugsmeridians liegen, sind die Verzerrungen der Abbildung relativ gering.

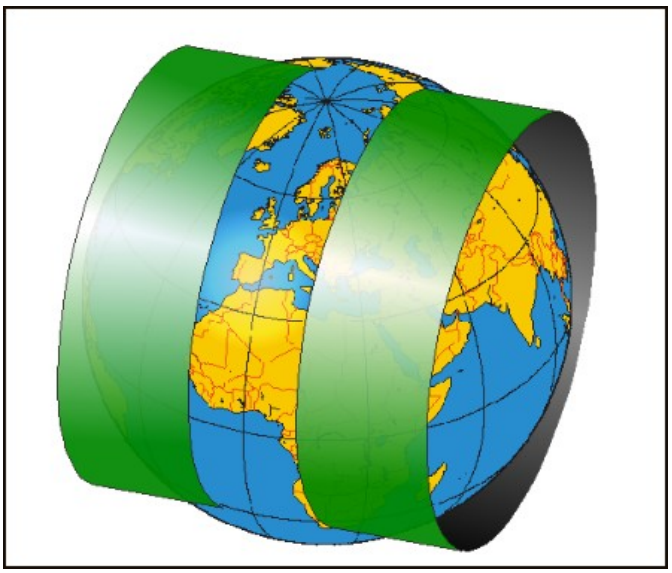


Abbildung 2.6: Transversale Zylinderprojektion der Erde

In der Praxis heißt dies, dass bei Gebieten, die nicht mehr als 1,5 Längengrade links oder rechts des genutzten Bezugsmeridians liegen, die Verzerrungen vernachlässigbar sind. Als Bezugsmeridiane werden Vielfache von 3 gewählt, also 0,3,6,9 bis 177. Der Ursprung des Koordinatensystems ist der Schnittpunkt zwischen dem Bezugsmeridian und dem Äquator. Der Wert auf der x-Achse des Koordinatensystems wird als Rechtswert (RW) bezeichnet, der auf der y-Achse als Hochwert (HW). Meistens wird zum eigentlichen Rechtswert ein fester Wert addiert, um negative Werte zu vermeiden. In Deutschland wird üblicherweise 500000 m hinzuaddiert. Des weiteren wird dem Rechtswert an der siebten Vorkommastelle noch eine weitere Zahl hinzugefügt, die anzeigt, welcher Bezugsmeridian verwendet wurde. Beim Bezugsmeridian 3° wäre dies die 1, bei 6° die 2, und so weiter. Beispielsweise käme ein Rechtswert von 4500000 aus einer Gauß-Krüger-Projektion mit Bezugsmeridian 12°. Die Einheit von Rechtswert und Hochwert ist dabei 1 Meter.

2.1.4 Überführung in internes Koordinatensystem

Für die Positionierung eines Objektes im Unity-eigenen Koordinatensystem werden zum einem der Rechtswert und Hochwert des Gauß-Krüger-Koordinatensystems verwendet und zu anderem die Höhe, die direkt aus den GPS-Koordinaten verwendet werden kann. GPS-Koordinaten bestehen meistens aus der Angabe der geographischen Länge und der geographischen Breite. Ein Wert für die geographische Breite wäre z.B. 48,75° N, wobei das N für north (Norden) steht. Das Gegenstück ist dementsprechend die Angabe von S

für south (Süden) bei Werten auf der Südhalbkugel. Eine alternative Methode ist das N und S wegzulassen und stattdessen bei Orten auf der Nordhalbkugel positive und bei Orten auf der Südhalbkugel negative Werte zu notieren. Analog ist die Schreibweise für den zweiten Wert, W (west) für Werte westlich des Nullmeridians, E (east) für Werte östlich des Nullmeridians. Die Höhe ist einfach die Angabe derselben in Meter. Die GPS-Koordinaten werden mit Hilfe eines entsprechenden Algorithmus in Rechtswert und Hochwert des Gauß-Krüger-Koordinatensystems umgerechnet. Dieser Algorithmus wurde von Herrn Prof. Zentgraf zu Verfügung gestellt (siehe auch [Kar]). Nach dieser Transformation noch sind folgende zwei Schritte notwendig, um die Überführung in das interne Koordinatensystem abzuschließen:

Ursprung anpassen: Das Anpassen des Ursprungs dient hauptsächlich der Vereinfachung. Der normale Koordinatenursprung liegt beim Schnittpunkt von Bezugsmeridian und Äquator, plus dem Versatz durch die Anpassung des Rechtswerts, welcher wie bereits oben erwähnt bei 500000 m liegt. Der Ursprung liegt damit bei 0° geographischer Breite, das betrachtete Gebiet (Mitteleuropa) aber mindestens bei 40° N geographischer Breite. Da der Ursprung daher weit außerhalb des relevanten Gebietes liegt, bietet sich für das interne Koordinatensystem eine Verschiebung des Ursprungs an. Am nächstliegendsten ist dabei, den linken unteren Rand des betrachteten Areals auf den Ursprung des Unity-Koordinatensystems zu setzen. Man subtrahiert einfach diesen Wert bei der Transformation von Gauß-Krüger in das interne Koordinatensystem.

Skalierung: Der Skalierungsfaktor ergibt sich durch das Festlegen von zwei Parametern. Der erste Parameter ist die Größe des Terrainstücks in der Realität, angegeben in Meter, das in Unity umgesetzt werden soll. Der zweite Parameter ist die Größe dieses Terrainstücks im Unity-Koordina"-tensystem. Ein Beispiel soll das ganze veranschaulichen. Die Seitenlänge des fraglichen Terrainstücks sei 25000 m groß, in Unity hat es die Ausmaße von 2048. Der Skalierungsfaktor s wäre also: $s = 2048/25000m = 0,08192m^{-1}$. Der Skalierungsfaktor wird mit dem Rechtswert, Hochwert und der Höhe multipliziert und zusammen mit dem Verschieben des Ursprungs, der natürlich auch mit dem Skalierungsfaktor verrechnet wird, ist die Transformation der Koordinaten ins Unity-interne Koordinatensystem vollständig.

Abb. 2.7 zeigt den Prozess der Koordinatentransformation in einer Gesamtübersicht.

Abbildung 2.7: Koordinatentransformation

2.2 Die Grafikengine Unity

2.2.1 Einführung

Um überhaupt etwas in einem Programm mit Hilfe von 3D-Grafik darstellen zu können, benötigt man Zugriff auf die Grafikschnittstelle des Rechners. Auf relativ niedriger Ebene der Schnittstellenabstraktion liegen dabei beispielsweise Direct3D oder auch OpenGL. Mit Hilfe von diesen Schnittstellen allein wäre es schon gut möglich, eine Simulation wie die in dieser Arbeit gefordert zu programmieren. Allerdings ist der Aufwand der Modellierung dabei noch recht hoch, da viele Objekte erst einmal mit Hilfe von einfacheren Funktionen zusammengebaut werden müssten. Weiter oben in der Abstraktionshierarchie liegen die Grafikengines. Diese bieten diverse Hilfsmittel, um die Modellierungen von 3D-Objekten einfacher zu machen und auch die Programmierung von Logik jenseits der puren Grafikdarstellung ist integriert. In Abb. 2.8 (nach [Ben06]) ist eine mögliche Schichteneinteilung für Computergrafik zu sehen.

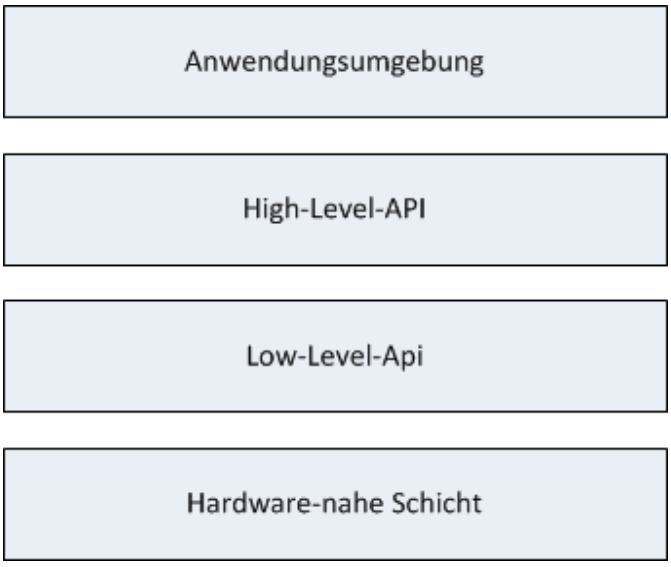


Abbildung 2.8: Schichtenmodell Grafikschnittstellen

Als Grundlage für die Simulation wird die Grafikengine Unity verwendet (siehe [Uni]). Vertrieben wird sie von der Firma Unity Technologies in der

aktuellen Version 3.5.5 (Stand September 2012). Für die Entwicklung von Software mit Unity sind zum einem eine kostenlose Standardversion als auch eine kostenpflichtige Proversion verfügbar. Erstere unterscheidet sich hauptsächlich durch das Fehlen von einigen grafischen Effekten und dem nicht ausschaltbaren Einblenden des Markenzeichens von Unity am Anfang des mit Unity erstellten Programms. Nativ ausführbare Programme können auf dem PC für Windows und Mac OS X, auf mobilen Geräten für Android und iOS und für diverse Spielekonsolen erstellt werden. Des weiteren gibt es die Möglichkeit, eine Webvariante zu erstellen, die mit einem entsprechenden Plugin in den häufigsten Browsern abgespielt werden kann. Verwendet wird Unity zum größten Teil als Grafikengine für Spiele. Aber auch für andere Anwendungsgebiete, wie z.B. wie für Simulationen bei dieser Arbeit, ist sie geeignet.

2.2.2 Überblick Arbeitsablauf

Im Allgemeinen ist der Unity Editor ähnlich aufgebaut wie viele andere Editoren und IDEs auch. Neben dem Hauptfenster, in dem hauptsächlich gearbeitet wird, sind noch diverse Toolleisten an den Seiten (siehe Abb. 2.9). Im Haupt-

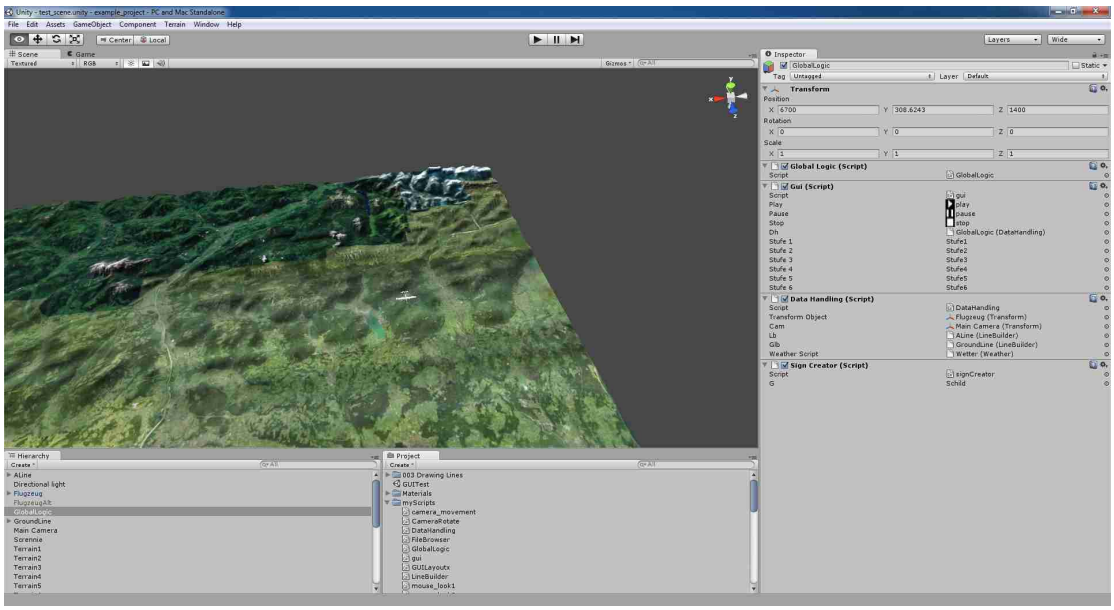


Abbildung 2.9: Ausschnit Unityeditor

fenster wird normalerweise die 3D Präsentation eben jener Szene angezeigt, an der man gerade arbeitet. Man sieht also wie es dann im Programm aussieht, abzüglich der erst nach dem Start des Programms geladenen Daten und dynamischem Ablauf. Sämtliche in einer Szene vorkommenden Objekte sind

auch als solche in einer Objekthierarchie vorhanden. Diese Objekte können untereinander hierarchisch untergliedert werden. Jedes Objekt kann mehrere sogenannte Komponenten beinhalten. Ausschließlich über diese wird auch die Darstellung im 3D-Raum realisiert. Standardmäßig gibt es eine große Anzahl dieser Komponenten, die über verschiedenste Parameter den eigenen Bedürfnissen angepasst werden können.

Neben den Komponenten sind Skripte das wichtigste Arbeitsmittel in Unity. Wie die Komponenten, müssen diese zwingend an ein Objekt gebunden sein. Trotzdem kann man von jedem Skript aus auch Funktionen ausführen, die globale Wirkung haben. Dies ist notwendig für Programmlogik, die sich auf einzelne Objekte beschränkt oder auch für Hilfsfunktionen wie z.B. diverse mathematische Operationen. Geschrieben werden können die Skripte in den Programmiersprachen C#, Javascript und Boo. Für diese Arbeit werden die Skripte mit C# erstellt. Die Skripte an sich werden nicht im Unityeditor geschrieben, sondern in der bereits mitgelieferten IDE MonoDevelop. Standardmäßig wird aber auch speziell für die Entwicklung mit C# Visual Studio auf Windowsrechnern unterstützt.

Das Verwalten von Builds für verschiedene Zielplattformen gestaltet sich recht einfach. Normalerweise wird die Zielplattform erst beim Erstellen der fertigen Version angegeben. Der Hauptunterschied liegt im unterstützten Funktionsumfang. Solange man nur Funktionen verwendet, die von allen gewünschten Zielplattformen unterstützt werden, kann man das Programm einfach für die entsprechende Zielplattform erstellen lassen.

Kapitel 3

Implementierung

3.1 Systemkonzept

Am Anfang jeder Überlegung steht bei diesem Projekt die Unterscheidung zwischen zwei Versionen des Programmes. Wie in 1.3 erwähnt soll es die Möglichkeit geben, einen Hagelabwehreinsatz live via Simulation mitverfolgen zu können. Dafür bietet es sich an, das Programm als Webplayervariante von Unity zur Verfügung zu stellen. Der Benutzer kann mit fast allen gängigen Browsern nach Installation eines Plugins auf das Programm zugreifen. Eine zweite Variante stellt eine allein lauffähige Version dar. Diese wird als normal ausführbares Programm erstellt und soll nur den Beteiligten von ROBERTA zur Verfügung stehen. Im Grunde sollen beide Varianten eine gleiche Basisfunktionalität aufweisen, allerdings hat die Webplayervariante folgende Einschränkungen:

- Darstellung beschränkt auf gerade stattfindende Hagelabwehreinsätze
- Keine Darstellung von Wetterdaten

Der Verzicht auf die Darstellung der Wetterdaten ist zwar sehr bedauerlich, allerdings aus rechtlichen Gründen unvermeidbar. Für zukünftige Versionen versucht die Hochschule ein Nutzungsrecht zu erhalten, denn die Wetterdarstellung würde einen erheblichen Mehrwert für den externen Benutzer bedeuten. In Tabelle 3.1 sind die Gemeinsamkeiten und Unterschiede der beiden Versionen zusammengefasst.

3.1.1 Externe Schnittstellen

Die vom Programm verarbeiteten Daten lassen sich anhand ihrer Quelle in drei Kategorien unterteilen. Im Folgenden sind dies:

Terraindaten: Alle Daten, die zur Simulation der Umgebung nötig sind. Damit soll es zum einem möglich sein, die Landschaft so darzustellen, dass

	Standalone	Webplayer
Livedarstellung	ja	ja
Wetterdarstellung	ja	nein
Abspielen alter Einsätze	ja	nein
Frei zugänglich	nein	ja

Tabelle 3.1: Gegenüberstellung der Programmversionen

jeder sich ohne große Probleme schnell zurecht finden kann. Zum anderen sollen aber die dafür benötigte Datenmenge nicht allzu groß sein.

Wetterdaten: Jene Daten, mit deren Hilfe das Wetter dargestellt werden kann. Hierbei kommt es besonders darauf an, dass man unterschiedlichen Intensitäten in den Unwettern erkennen kann.

Flugzeugdaten: Alle Informationen die während des Fluges gesammelt werden. Diese werden für die Simulation des Fluges benötigt. Auch hier werden Daten über das Wetter gesammelt, die aber völlig anderer Art wie die vorher genannten Wetterdaten sind und auch anders dargestellt werden.

Abb. 3.1 stellt einen Überblick über das System und seine Komponenten dar.

Anbindung an die Datenbank

Die Anbindung an die FH-interne Datenbank, in der sowohl die Daten des Fliegers als auch die des Wetters gespeichert sind, ist von besonderer Bedeutung, da zwei der drei Datenquellen darüber ausgelesen werden können. Verwendet wird dafür eine mySQL-Datenbank (siehe [MYS]). Für einen direkten Zugriff auf eine solche existieren für das .net-Framework vorgefertigte Lösungen, die auch mit diesem Programm verwendet werden können. Ein solcher ist der MySQL Connector/Net (siehe [Mon]). Ein erster Versuch damit hat nach kleineren Anpassungen problemlos funktioniert. Allerdings hat diese Lösung zwei gravierende Mängel. Zum einem liegt die Datenbank im internen Netz der Fachhochschule. Der Benutzer bräuchte also einen Zugang zu dem Netzwerk. Eine Offenlegung der Daten in einer frei zugänglichen Datenbank kommt auch nicht in Frage. Zum anderen gibt es eine technische Beschränkung. So funktioniert der direkte Zugriff auf die Datenbank nicht mit der Webplayervariante. Für diese muss also ohnehin eine andere Lösung gefunden werden. Eine solche ist in Abb. 3.2 zu sehen.

Diese Methode folgt dem von Unity vorgeschlagenen Weg, Daten bei der Webplayerversion zu erhalten, nämlich über einen http-Request. Um weiteren

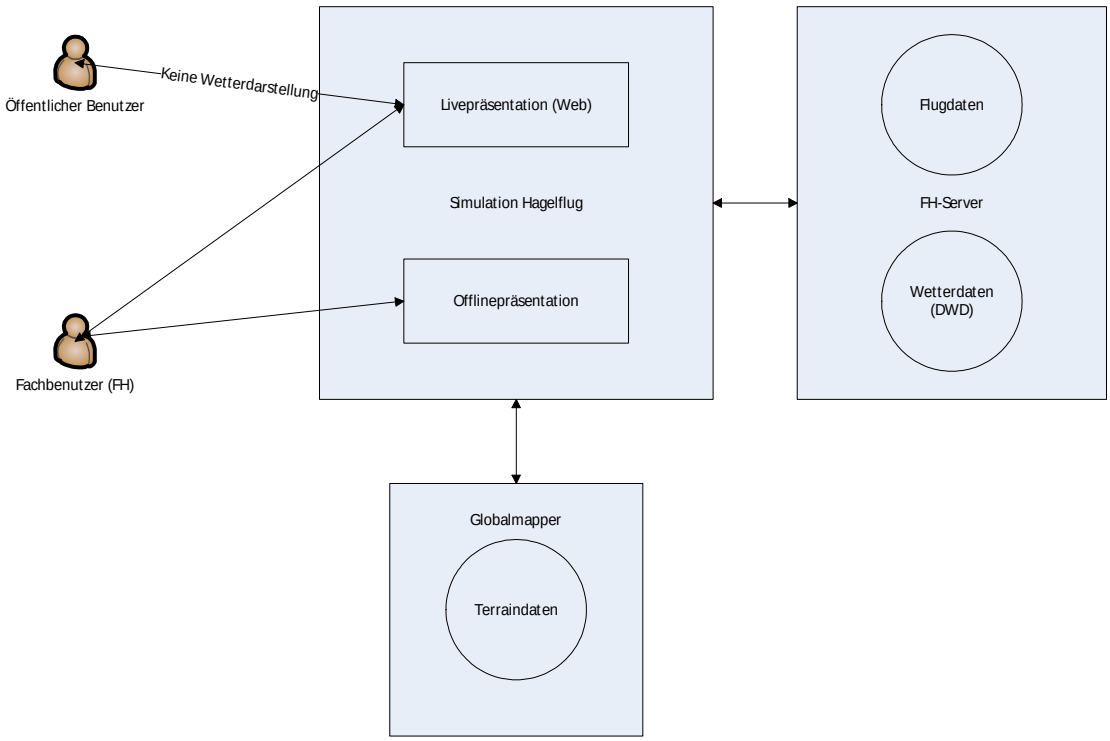


Abbildung 3.1: Übersicht Gesamtkonzept

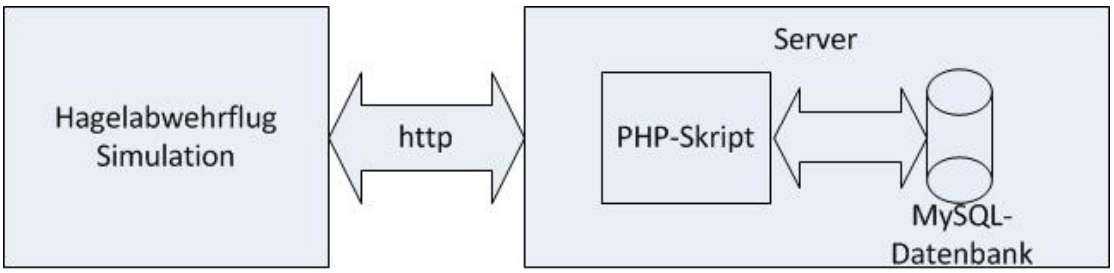


Abbildung 3.2: Anbindung an Datenbank

Aufwand zu vermeiden, wird diese Art der Datenübertragung bei der Standalone Variante ebenfalls benutzt. Zum Start der Datenübertragung sendet das Programm einen http-Request an den Rechner, auf dem auch die Datenbank liegt. Bei der Webplayerversion liegt das Programm ebenfalls auf diesem Rechner, wodurch die Anfrage keine weiteren Schritte erfordert. Bei der anderen Variante muss der Benutzer mit dem internen Netzwerk der Hochschule verbunden sein, damit der http-Request den Zielrechner erreicht. Der http-Request wird von einem php-Skript weiterverarbeitet. Dieses kümmert sich um den Datenbankaufruf. Je nach dem http-Request hinzugefügten Parame-

tern schickt das php-Skript die entsprechenden Daten an das Simulationsprogramm.

3.1.2 Interne Logik

Wie bereits in Kapitel 2.2.2 erwähnt, sind die Skripte das Herzstück der Programmlogik. Diese erben standardmäßig von der Klasse MonoBehaviour. Diese stellt insbesondere zwei Funktionen zur Verfügung, die essentiell für die Laufzeitsteuerung des Programmes sind.

Start(): Diese Funktion wird automatisch und einmalig bei der Initialisierung des zugehörigen Skriptes aufgerufen. Typischerweise werden hier alle Funktionen aufgerufen und Daten initialisiert, die für das Arbeiten mit dem Objekt am Anfang des Programmes notwendig sind.

Update(): Diese Funktion wird immer dann aufgerufen, wenn ein neuer Frame erzeugt wird. Unter einem Frame versteht man hierbei die Einzelbilder, aus dem die Simulation aufgebaut ist. Durch das Zusammensetzen der Frames hintereinander entsteht für den Betrachter die dynamische Darstellung. Diese Funktion wird während der gesamten Laufzeit des Programmes bei jedem neuen Frame und damit immer wieder aufgerufen. Das macht diese Funktion zum idealen Ort für Code, der unabhängig von anderen Eingaben immer wieder aufgerufen werden muss.

Die Update()-Funktion ist auch die Funktion, die in der Simulation genutzt wird, um Programmcode einen fest definierten Zeitraum lang laufen zu lassen. Ist es z.B. nötig, den Code nach einem gewissen Zeitschritt immer wieder aufzurufen, so kann dazu die interne Zeitfunktion von Unity genutzt werden. Ist die Differenz zwischen aktuellem Zeitpunkt und dem Zeitpunkt seit dem letzten Aufruf des Codes größer als der eingestellte Zeitschritt, wird der Code erneut aufgerufen. Diese Funktionalität wird z.B. bei der Bewegung des Flugzeuges in der Simulation genutzt. Durch das Einstellen der Größe des Zeitschritts kann man die Bewegung des Flugzeuges verlangsamen bzw. beschleunigen.

3.2 Datenquellen

3.2.1 Daten Terrain

Von allen zu zuführenden Daten ist bei den Terraindaten am meisten Vorarbeit zu leisten. Im Gegensatz zu den anderen Daten werden diese nicht in irgendeiner Form von der Hochschule zur Verfügung gestellt. Sie müssen also

aus einer anderen Quelle importiert werden. Auch gibt es keine verwendbaren Erfahrungswerte aus [Hög09], da bei dieser Arbeit alles was mit dem Gelände und seiner Darstellung zusammenhängt, von Google Earth erledigt wird. Als erstes stellt sich die Frage, in welcher Art und Form man die Terraindaten bezieht und aus welcher Quelle.

Am günstigsten stellt sich die Zweiteilung der Terraindaten heraus. Dies wären im Folgenden:

- Höhendaten
- Satellitendaten

Die Höhendaten beinhalten alle geographisch wichtigen Daten. Wie der Name schon andeutet, enthalten sie ausschließlich Informationen über die Höhe des Geländes (Beispiel siehe Abb. 3.3).

780,09 m	768,31 m	750,47 m	720,90 m	699,54 m
791,67 m	777,32 m	761,80 m	744,77 m	699,99 m
801,56 m	789,46 m	755,60 m	749,07 m	722,39 m
830,94 m	800,39 m	788,33 m	766,99 m	757,41 m
822,48 m	804,73 m	789,11 m	772,89m	767,67 m

Abbildung 3.3: Beispielhafte Darstellung der logischen Datenstruktur der Höhendaten

Informationen über den Referenzpunkt der Höhendaten im Gelände und den Abstand zwischen zwei nebeneinanderliegenden Datenpunkten, ergo die Auflösung, sind in den Dateien nicht vorhanden. Sie wurden während des Extrahierens der Daten bestimmt und müssen beim Importieren der Daten nach Unity bekannt sein und dementsprechend angepasst werden.

Die Satellitendaten repräsentieren die für die grafische Darstellung der Landschaft nötigen Daten. Im Normalfall sind dies Satellitenfotos der Erde. Durch die Zweiteilung der Daten ist es allerdings möglich, auch andere Dinge darzustellen als reine Fotographien der Landschaft. Eine Möglichkeit wäre z.B. eine Straßenkarte zu verwenden. Dies kann unter Umständen zu einem schnelleren Überblick über die Landschaft führen, da große Orte und Straßen besser erkennbar sind.

Darzustellendes Gebiet

Die Größe der Landschaft, die im Programm simuliert werden soll, definiert sich über die Region, von der maximal Wetterdaten des deutschen Wetterdienstes (DWD) erhoben werden (siehe dazu Tabelle 3.5). Dies entspricht einem Areal von mindestens 400x400 km. Aber um fehlende Gebiete durch die

unterschiedlichen Arten der Projektionen zu vermeiden, wurde ein Gebiet von 450x450 km gewählt (siehe Abb. 3.4). Damit umfasst das Areal mehrere Längengrade, die als Bezugsmeridiane für die Gauß-Krüger-Projektion verwendet werden können. Am günstigen ist dabei die Wahl von Längengrad 12, da er von allen Bezugsmeridianen am zentralsten liegt. Zwar sind durch die Größe des Gebietes die Verzerrungen am Rand etwas größer als bei der Wahl einer einzelnen Projektion, aber für die Anwendung sind diese nicht von Bedeutung. Für echte Auswirkungen müssten die Verzerrungen mindestens Größenordnungen von 100 m und mehr betragen, was hier nicht der Fall ist.

Abbildung 3.4: Darzustellendes Gebiet in der Simulation

Die Eckpunkte des Terrains sind, in Gauß-Krüger-Koordinaten, in Tabelle

Links oben RW: 4250000 HW: 5575000	**Rechts oben** RW: 4700000 HW: 5575000
Links unten RW: 4250000 HW: 5125000	**Rechts unten** RW: 4700000 HW: 5125000

Tabelle 3.2: Eckpunkte des dargestellten Terrains

3.2 zu sehen.

Rasterisierung

Neben der Zweiteilung der Datensätze ist die Rasterisierung ebenfalls ein wichtiger Aspekt. Damit ist hier die Tatsache gemeint, dass das aus dem abzubildenden Gebiet nicht nur jeweils eine Höhendatei und Satellitendatei erzeugt wird, sondern mehrere. Zwar bedeutet dies an vielen Stellen einen größeren Verwaltungsaufwand, aber es ergeben sich auch einige Vorteile und an anderen Stellen ist die Rasterisierung sogar unumgänglich, wie in Kapitel 3.3.1 näher erläutert wird. Abb. 3.5 veranschaulicht das Prinzip der Rasterisierung.

Die Stufe der Rasterisierung ist dabei so zu wählen, dass ein gutes Gleichgewicht zwischen Aufwand und gewonnener Funktionalität entsteht. Eine gute Einteilung kann dabei nur durch Ausprobieren ermittelt werden. Erwähnenswert ist auch noch, dass theoretisch die Rasterisierung in x- und y-Richtung jeweils unterschiedlich gewählt werden kann. Da aber das dargestellte Gebiet quadratische Ausmaße hat und quadratische Teilstücke leichter zu verwalten sind, wird immer in beiden Richtungen die selbe Rasterisierung verwendet.

Datenquelle und Datenverarbeitung

Nach den Überlegungen über die Art der Daten muss eine Datenquelle gefunden werden, aus der die Daten gewonnen werden können. Als Programm, mit dem man sowohl die Höhendaten als auch die Satellitendaten extrahieren kann, kam Global Mapper zum Einsatz (siehe [Glo]). Bei Global Mapper handelt es sich um ein Geoinformationssystem. Mit diesem kann man auf verschiedenste Geodaten zugreifen und sie in Form von anderen Formaten

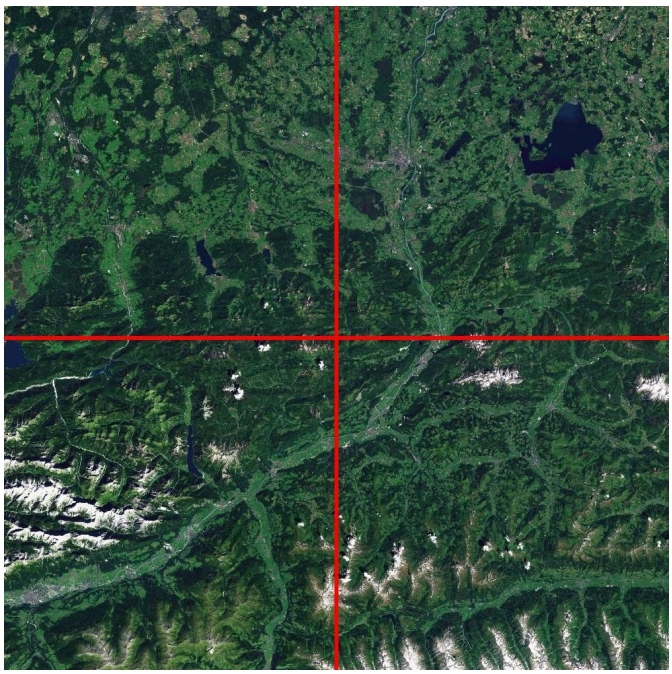


Abbildung 3.5: Rasterisierung Terrain

exportieren. Für die Höhendaten ist die Datenverarbeitungskette in Abb. 3.6 zu sehen.

Die Grundlage für die Höhendaten bildet ASTER GDEM Worldwide Elevation Data[1] Datenbank. Auf diese greift man mit Hilfe von Global Mapper zu. Nachdem man noch die Projektionsart auf Gauß-Krüger umgestellt hat, muss man nur noch das Dateiformat und die Ausmaße des zu extrahierenden Gebietes auswählen. Auch die weiter oben erwähnte Rasterisierung kann mit beliebig vielen Stufen ausgewählt werden. Als Exportformat wurde das ter-Format ausgewählt. Dieses lässt sich ohne Probleme mit dem nächsten Programm in der Verarbeitungskette laden. Dabei handelt es sich um das Programm L3DT (siehe [L3D]). Mit diesem werden werden die eingelesenen ter-Dateien wiederum in das raw-Format exportiert. In L3DT lassen sich die minimale und maximale Höhe des Geländes auslesen, was für das Anpassen der Höhe der Terrainstücke zueinander wichtig ist, wie in Kapitel 3.3.1 näher erläutert wird. An dieser Stelle besteht keine Wahlfreiheit bei der Art des Dateiformates, da Unity nur raw-Dateien für die Terraindarstellung benutzen kann.

Für das Extrahieren der Satellitenbilder kommt ebenfalls Global Mapper zum Einsatz. Als Datenquelle wird die Datenbank namens World Imagery[2] verwendet. Analog zu den Höhendaten erfolgt auch hier erst die Anpassung

1 http://asterweb.jpl.nasa.gov/gdem.asp
2 http://www.arcgis.com/home/item.html?id=10df2279f9684e4a9f6a7f08febac2a9

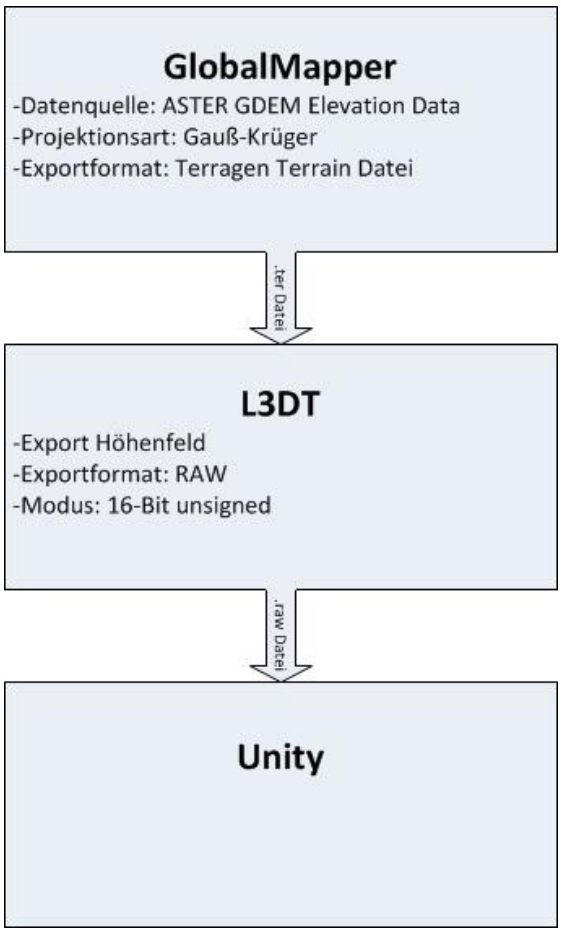


Abbildung 3.6: Datenverarbeitungskette Höhendaten

der Projektionsart und dann das Festlegen der Parameter des zu extrahierenden Gebietes und die Stufe der Rasterisierung. Die Satellitenbilder werden in Form von png-Bildern gespeichert. An dieser Stelle ist keine weitere Verarbeitung der Daten nötig, weil die png-Dateien als solche direkt von Unity verwendet werden können. Abb. 3.7 veranschaulicht das Ganze noch einmal.

3.2.2 Daten Hagelflieger

Bei diesen Daten handelt es sich um alle Daten, die während eines Hagelabwehrfluges gesammelt wurden. Die Daten werden von insgesamt drei Messsystemen erfasst, von denen sich zwei jeweils im rechten und linken Flügel und eines im vorderen Teil des Flugzeuges befinden. Die Systeme in den Flügeln sind vom Typ HAILsens und das dritte vom Typ HAILaquisition (siehe [Ber12a] und [Fre11]). Der Hauptunterschied zwischen HAILsens und HAILa-

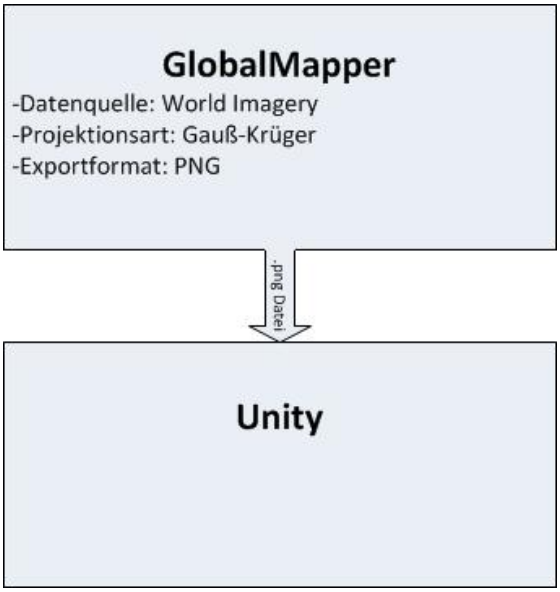


Abbildung 3.7: Datenverarbeitungskette Satellitendaten

quisition ist der, dass nur HAILaquisition über GPS-Ortung verfügt. Tabelle 3.3 zeigt die Art der Daten auf, die von allen Systemen erfasst werden, plus die GPS Daten aus HAILaquisition. Eine Tabellenzeile repräsentiert dabei genau einen Sensor des Messsystems.

Die wichtigsten Daten sind dabei die Positionsdaten. Ohne diese ist es nicht möglich, eine Nachbildung des Hagelabwehreinsatzes in der Simulation zu realisieren. Auch die Informationen zur Uhrzeit und die zur Lage im Raum werden für die vollständige Nachbildung des Fluges benötigt. Neben diesen Daten werden allerdings auch noch weitere Daten während des Fluges aufgezeichnet, wie z.B. Temperatur, Luftdruck oder auch der aktuelle Silberi-

Messgröße	**Beschreibung**
Luftdruck	relativer Luftdruck in hPa
Temperatur und Luftfeuchtigkeit	Temperatur in °C, relative Luftfeuchtigkeit in %
Ausrichtung, Nicken, Rollen, Magnetfeld in x-, y-, und z-Richtung	Angaben des elektronischen Kompasses, des Nickens und des Rollens in °, Magnetfeld in µT
Beschleunigung und Drehraten in x-, y- und z-Richtung	Beschleunigung in m/s^2, Drehraten in °/s
GPS Daten	Länge, Breite, Höhe

Tabelle 3.3: Gemessene Daten der Messsysteme im Flugzeug

odidausstoß. Die ganzen erfassten Daten werden mit Hilfe eines während des RO-BERTA Projektes entwickelten Protokolls während des Fluges an einen Rechner geschickt, der in der FH Rosenheim steht. Zusätzlich werden aber alle Messdaten der Sensoren noch auf einem extra Datenträger aufgezeichnet, der sich im Flugzeug selbst befindet. Somit ist sicher gestellt, dass im Falle von Funkunterbrechungen während eines Hagelabwehreinsatzes trotzdem alle Messdaten zur späteren Auswertung und Weiterverarbeitung zur Verfügung stehen.

Die Positionsdaten werden in Form von GPS-Koordinaten übergeben. Um sie im programminternen Koordinatensystem zu verwenden, werden sie zuerst in das Gauß-Krüger-Koordinatensystem transformiert. Jetzt müssen die Positionsdaten nur noch entsprechend eines vorher gewählten Koordinatenursprunges verschoben und mit dem richtigen Skalierungsfaktor skaliert werden. Wie dies genau funktioniert, wurde bereits in Kapitel 2.1.4 erläutert.

Die restlichen Daten jenseits der Positionsdaten werden zwar nicht für die Simulation des Fluges benötigt, dennoch beinhalten sie informative Angaben. Wie diese dann im Programm dargestellt werden können, damit beschäftigt sich Kapitel 3.3.2

3.2.3 Daten Wetter

Die Daten zum Wetter stammen direkt vom deutschen Wetterdienst. Im Grunde handelt es sich bei diesen Wetterdaten um Radarmessungen (siehe [DWD]). Die hier verwendete Radarstation ist eine vom DWD in der Nähe des Flughafens München betriebene Station. Somit ist dies auch das Zentrum der Radarmessungen. Vereinfacht gesagt wird die Echostärke des Radarsignals gemessen. Der Grad der Reflexion des Radarsignals ist abhängig von der Menge des Wassers in der wasserführenden Luftschicht und auch von der Größe der Wassertropfen. So lässt sich eine direkte Korrelation zwischen dem Echosignal und der Stärke des Niederschlages feststellen. Ist der gemessene Wert niedrig, kann man mit leichtem Niederschlag rechnen. Ist der Wert allerdings am oberen Ende der Messskala, so ist an dieser Stelle starker Niederschlag möglich, z.B. Starkregen oder eben Hagel. Die Bezeichnungen gehen dabei, neben dem von keinem nennenswerten Messwert, von sehr leichtem bis zu extremen Niederschlag. In der Radarstation selbst werden dabei verschiedenste Messungen mit unterschiedlichen Auflösungen und Ausmaßen angefertigt. Für diese Arbeit von Interesse sind dabei nur die sogenannten PX- und PZ-Daten.

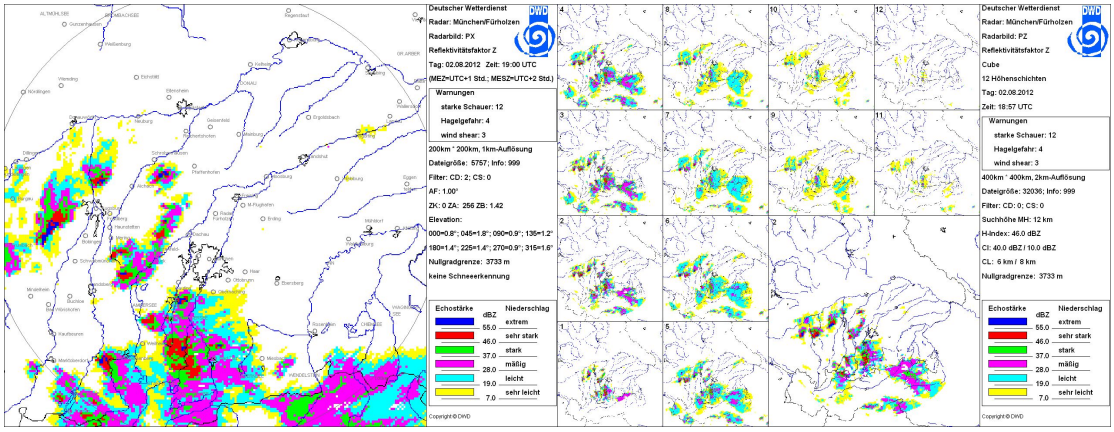
Die PZ-Daten stellen ein Gebiet von ca. 200 km um die Radarstation dar; also ein Gebiet von 400x400 km. Die horizontale Auflösung beträgt ungefähr 2 km. Vertikal werden insgesamt 12 Schichten gemessen, wobei die niedrigste in einer Höhe von 1 km liegt und jede weitere Schicht einen 1 km weiter oben,

ergo ein Bereich 1000-12000 m. Die Messungen finden alle 15 Minuten statt.

Die PX-Daten dagegen stellen ein Gebiet von ca. 100 km um die Radarstation dar, was dann ein Gebiet von 200x200 km ergibt. Die horizontale Auflösung beträgt ca. 1 km. Gemessen wird im Gegensatz zu PZ nur eine Schicht, nämlich die unterste Schicht bei 1 km Höhe. Die Messungen erfolgen alle 5 Minuten.

Zusätzlich ist noch zu erwähnen, dass alle Radarinformationen auf einer Karte mit polarstereographischer Projektion gewonnen wurden. Dies hat zur Folge, dass das Koordinatensystem der Radardaten und das von Unity, was mit Hilfe von Gauß-Krüger-Projektion gewonnen wurde, nicht exakt zueinander passen. Da die Daten aber mindestens einen Abstand von 1 km haben und die Ungenauigkeiten bei dem verwendeten Gebiet höchstens wenige Meter ausmachen, wurde hier auf eine Korrektur verzichtet.

In Abb. 3.8 sieht man eine grafische Darstellung der PX- und PZ-Daten von Seiten des DWD vom 02.08.2012 um ca. 19 Uhr.



(a) Darstellung PX (b) Darstellung PZ

Abbildung 3.8: Wettersituation 02.08.2012 um 19 Uhr

Die logische Datenstruktur ist für beide Datentypen ähnlich. Die wörtlichen Beschreibungen des Niederschlags von kein, einfacher bis zu extremer Niederschlag werden als Zahlenwerte repräsentiert. Diese Zahlenwerte gehen dann von 0 für kein Niederschlag bis 6 für extremer Niederschlag. Für PX und PZ ergeben sich für das dargestellte Gebiet und die örtliche Auflösung jeweils 200 Werte in x-Richtung und y-Richtung. Eine Schicht weist somit 40000 Messdaten auf. Bei PZ liegen dann dementsprechend wegen der 12 gemessenen Schichten 480000 Messwerte vor. Da die logische Struktur der Daten für die graphische Darstellung wichtig ist, soll Abb. 3.9 die Strukur einer Schicht aufzeigen.

Die Hochschule hat im Zuge von RO-BERTA erreicht, dass sie vom DWD regelmäßig die von der Radarstation München gemessenen PX- und PZ-Daten

```
1;0;0;0;0;0;0;0;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
1;1;0;0;0;0;0;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
2;2;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;0;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
2;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;2;0;0;0;0;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;1;1;0;0;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
1;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;1;2;2;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
1;1;1;0;0;0;0;0;0;0;0;0;1;0;1;0;0;0;0;1;0;1;1;2;2;2;1;0;0;0;0;0;0;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
1;1;1;0;0;0;0;0;0;0;0;1;1;1;2;1;1;0;1;1;1;1;1;2;2;2;1;0;0;0;0;0;0;2;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
1;1;1;0;0;0;0;0;0;1;1;1;1;1;2;1;2;1;2;1;2;1;1;1;1;1;1;0;1;2;1;2;3;3;2;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
1;1;1;0;0;0;0;0;0;1;1;1;1;1;1;2;1;1;2;2;2;2;1;1;1;1;1;0;1;2;2;2;2;2;1;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
1;1;1;0;0;0;0;0;0;1;1;1;1;1;1;1;1;2;2;3;3;3;4;2;2;2;2;2;2;2;1;2;2;2;1;1;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
1;1;1;0;0;0;0;0;0;1;1;1;1;2;1;1;1;2;3;3;3;3;4;4;4;3;3;3;2;1;1;2;2;2;2;1;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
1;1;1;0;0;0;0;0;0;1;1;1;2;2;1;2;2;2;3;3;3;3;4;5;4;4;4;3;1;0;1;2;3;2;1;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;
```


Abbildung 3.9: Arraydarstellung einer Wetterschicht

erhält. Diese liegen zunächst in einem speziellen binären Format vor. Mit Hilfe eines Perlskriptes kann dieses dann in das .csv Format umgewandelt werden. Diese Datei kann dann leicht ausgelesen werden, da die Daten exakt, wie weiter oben besprochenen, in logischer Datenstruktur vorliegen. Die PX-Datei ist damit im Endeffekt ein 2-dimensionales Array, die PZ-Datei dann ein 3-dimensionales Array bzw. 12 2-dimensionale Arrays. Das Einlesen dieser csv-Dateien wurde im Zuge dieses Projektes erstmals für das Einlesen von Wetterdaten realisiert.

Die Wetterdaten sollen aber nicht nur als lose Einzeldateien auf dem Rechner liegen, sondern auch in einer Datenbank. In dieser liegen dann die Daten in einer anderen Form vor wie bei der csv-Datei. Dies geschieht hauptsächlich in Hinsicht auf die Speicherplatzoptimierung. Würden die Daten wie bei der csv-Variante als 2-dimensionales bzw. 3-dimensionales Array gespeichert, würden alle Messwerte gespeichert. Aber insbesondere an niederschlagsfreien Tagen haben die gemessenen Werte überwiegend den Wert 0. Diese sind uninteressant für die Niederschlagsauswertung und haben auch keine grafische Repräsentation in der Wettergrafik des DWD. Stattdessen sind in der Datenbank für die jeweiligen Intensitäten von 1 bis 6 die Positionen in dem Array angegeben, wo ein solcher Wert vorkommt (siehe Abb. 3.10). Für die Simulation ist dies insofern von Bedeutung, dass die Daten anders eingelesen werden müssen als bei der Variante mit den csv-Dateien.

3.3 Grafische Darstellung

3.3.1 Terrain

Die Darstellung des Terrains in der Simulation ist der erste Schritt bei der Erstellung der virtuellen Umgebung. Ohne diese ist nur schwer schnell und intuitiv nachzuvollziehen, wo ein Hagelabwehrflug stattfindet. Da es, wie in Kapitel 3.2.1 bereits dargestellt, zwei verschiedene Datensätze gibt, wird die

ID	FileDateAcqu	FileTimeAcqu	REP_Class6	REP_Class5	REP_Class4	REP_Class3	REP_Class2	REP_Class1
1	120802	1512	[BLOB - 36,0KiB]	[BLOB - 34,0KiB]	[BLOB - 7,9KiB]	[BLOB - 2,0KiB]	[BLOB - 276Bytes]	[BLOB - 12Bytes]
2	120802	1527	[BLOB - 35,4KiB]	[BLOB - 33,3KiB]	[BLOB - 9,7KiB]	[BLOB - 2,7KiB]	[BLOB - 513Bytes]	[BLOB - 33Bytes]
3	120802	1542	[BLOB - 33,2KiB]	[BLOB - 29,8KiB]	[BLOB - 10,2KiB]	[BLOB - 2,4KiB]	[BLOB - 546Bytes]	[BLOB - 0Bytes]
4	120802	1557	[BLOB - 33,9KiB]	[BLOB - 26,9KiB]	[BLOB - 9,1KiB]	[BLOB - 2,4KiB]	[BLOB - 636Bytes]	[BLOB - 6Bytes]
5	120802	1612	[BLOB - 32,7KiB]	[BLOB - 27,7KiB]	[BLOB - 8,1KiB]	[BLOB - 2,8KiB]	[BLOB - 543Bytes]	[BLOB - 9Bytes]
6	120802	1627	[BLOB - 36,3KiB]	[BLOB - 28,0KiB]	[BLOB - 9,4KiB]	[BLOB - 3,3KiB]	[BLOB - 696Bytes]	[BLOB - 63Bytes]
7	120802	1642	[BLOB - 41,1KiB]	[BLOB - 28,6KiB]	[BLOB - 9,8KiB]	[BLOB - 3,4KiB]	[BLOB - 834Bytes]	[BLOB - 12Bytes]
8	120802	1657	[BLOB - 43,3KiB]	[BLOB - 33,7KiB]	[BLOB - 10,7KiB]	[BLOB - 4,4KiB]	[BLOB - 1,2KiB]	[BLOB - 72Bytes]
9	120802	1712	[BLOB - 49,9KiB]	[BLOB - 37,8KiB]	[BLOB - 12,7KiB]	[BLOB - 4,7KiB]	[BLOB - 1,5KiB]	[BLOB - 204Bytes]
10	120802	1727	[BLOB - 56,4KiB]	[BLOB - 42,1KiB]	[BLOB - 14,7KiB]	[BLOB - 5,7KiB]	[BLOB - 1,6KiB]	[BLOB - 189Bytes]
11	120802	1742	[BLOB - 64,2KiB]	[BLOB - 45,3KiB]	[BLOB - 14,2KiB]	[BLOB - 5,3KiB]	[BLOB - 1,7KiB]	[BLOB - 303Bytes]
12	120802	1757	[BLOB - 69,2KiB]	[BLOB - 50,4KiB]	[BLOB - 16,6KiB]	[BLOB - 5,5KiB]	[BLOB - 1,6KiB]	[BLOB - 222Bytes]
13	120802	1812	[BLOB - 73,6KiB]	[BLOB - 50,1KiB]	[BLOB - 17,6KiB]	[BLOB - 6,0KiB]	[BLOB - 1,8KiB]	[BLOB - 288Bytes]
14	120802	1827	[BLOB - 77,4KiB]	[BLOB - 51,1KiB]	[BLOB - 19,0KiB]	[BLOB - 6,2KiB]	[BLOB - 2,0KiB]	[BLOB - 396Bytes]
15	120802	1842	[BLOB - 75,9KiB]	[BLOB - 53,8KiB]	[BLOB - 19,3KiB]	[BLOB - 6,0KiB]	[BLOB - 2,0KiB]	[BLOB - 441Bytes]
16	120802	1857	[BLOB - 80,1KiB]	[BLOB - 56,0KiB]	[BLOB - 19,3KiB]	[BLOB - 5,3KiB]	[BLOB - 2,0KiB]	[BLOB - 510Bytes]
17	120802	1912	[BLOB - 85,4KiB]	[BLOB - 56,6KiB]	[BLOB - 19,0KiB]	[BLOB - 5,4KiB]	[BLOB - 1,5KiB]	[BLOB - 312Bytes]
18	120802	1927	[BLOB - 94,8KiB]	[BLOB - 57,0KiB]	[BLOB - 17,9KiB]	[BLOB - 5,3KiB]	[BLOB - 1,6KiB]	[BLOB - 432Bytes]
19	120802	1942	[BLOB - 96,6KiB]	[BLOB - 52,8KiB]	[BLOB - 17,0KiB]	[BLOB - 4,3KiB]	[BLOB - 1,6KiB]	[BLOB - 486Bytes]
20	120802	1957	[BLOB - 81,4KiB]	[BLOB - 25,5KiB]	[BLOB - 6,3KiB]	[BLOB - 2,5KiB]	[BLOB - 897Bytes]	[BLOB - 78Bytes]
21	120802	2012	[BLOB - 94,1KiB]	[BLOB - 33,7KiB]	[BLOB - 10,6KiB]	[BLOB - 3,9KiB]	[BLOB - 1,1KiB]	[BLOB - 93Bytes]
22	120802	2027	[BLOB - 105,8KiB]	[BLOB - 51,4KiB]	[BLOB - 17,6KiB]	[BLOB - 6,1KiB]	[BLOB - 1,2KiB]	[BLOB - 96Bytes]

Abbildung 3.10: Beispiel Datensatz aus Wetterdatenbank

Überführung in den 3D-Raum jeweils einzeln betrachtet.

Höhendaten

Die vorhandenen Höhendaten müssen als nächstes nach Unity importiert werden. Dies ist problemlos möglich, da sie nach den Maßnahmen aus Kapitel 3.2.1 bereits im richtigen Format vorliegen. Ein Terrainobjekt wird im Programm erstellt und nach Anpassung von Höhe, Breite und Länge ist nach Einlesen der .raw Datei auch schon ein Höhenprofil in der 3D-Simulation sichtbar (siehe Abb. 3.11).

Ein Problem beim Zusammenfügen der einzeln Terrainstücke gibt es allerdings. Beim Einlesen der .raw-Datei verwendet Unity den kleinsten Höhenwert als Nullreferenz. Dies hat zur Folge, dass zwischen den einzelnen Terrainteilen kein fließender Übergang existiert. Die .raw Datei jedes Einzelterrain hat einen eigenen kleinsten Höhenwert und vor allem zwischen Flachlandregionen und Gebirgsregionen ist dieser sehr unterschiedlich. Die Nullreferenz müsste bei allen Terrainteilen identisch sein, damit sie von Anfang an zusammenpasst. Um trotzdem eine schöne Darstellung der Landschaft zu ermöglichen, kann man auf einen Trick ausweichen. Man verschiebt die Terrainteile anhand der Minimalhöhe, angepasst an den Skalierungsfaktor in Unity, in der zugehörigen .raw-Datei in der Höhe. Im Endeffekt erreicht man damit, dass alle Terrainstücke beim Nullmeter ihre gemeinsame Basisreferenz haben. Die notwendigen Werte der kleinsten Höhenwerte werden vor dem Umwandeln in .raw-Dateien ausgelesen. Die Übertragung der ermittelten Werte nach Unity erfolgt manuell. Optimalerweise liest man die Werte mit Hilfe von Unity via

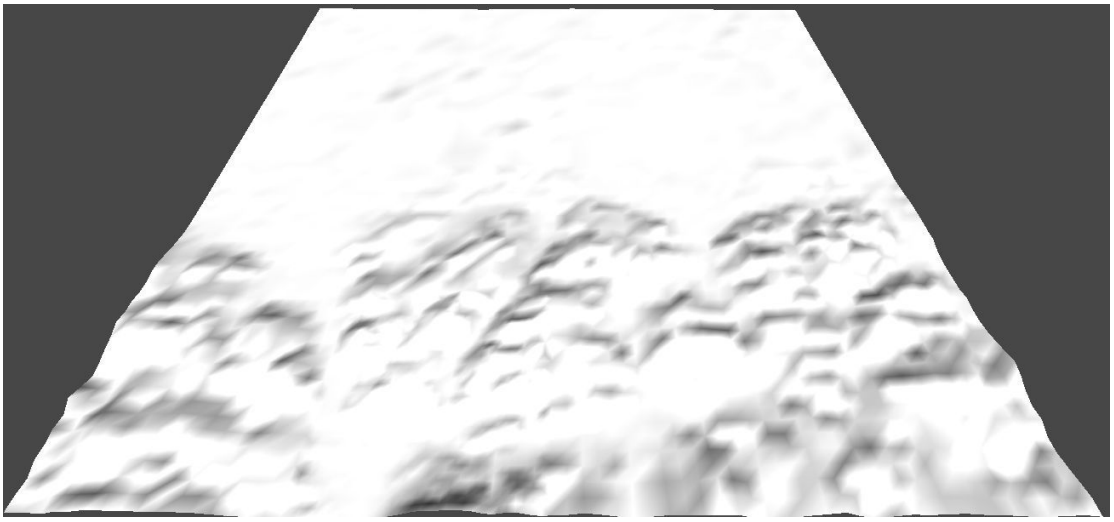


Abbildung 3.11: Höhenprofil 3D-Landschaft

Programmcode automatisch aus den Dateien, dies wurde aber auf Grund von Zeitmangel und Priorisierung anderer Funktionen nicht realisiert.

Texturdaten

Nachdem das Höhenprofil in Unity erstellt worden ist, folgt als Nächstes die Integration der Satellitendaten. Dazu werden die Satellitenbilder als erstes nach Unity importiert. Dann werden sie als Texturen dem zugehörigen Terrainobjekt hinzugefügt. Abb. 3.12 zeigt ein Terrainstück mitsamt dem Satellitenbild.

Wie bereits in Kapitel 3.2.1 erwähnt, können statt Satellitenfotos auch andere Objekte als Textur für das Terrain dienen, wie die Darstellung der Straßen und Wege. Es stellt sich also die Frage, wie diese anderen Texturobjekte in der Simulation verwendet werden können. Denn beim Start des Programmes sollte sinnvollerweise bereits eine Textur geladen worden sein, da ansonsten die Höhendaten einfach nur weiß wären, wie es in Abb. 3.11 zu sehen ist. Am geeignetsten dafür sind wohl die Satellitenbilder, da damit am Beginn ein möglichst realistische Abbildung der Wirklichkeit erreicht wird. Eine Möglichkeit andere Texturdaten in der Simulation zu ermöglichen, wäre z.B. die Option während der Laufzeit anzubieten, diese statt der Satellitenbilder auf die Höhendaten anzuwenden. Bei Auswahl dieser Option würden dann einfach alle Terrainobjekte mit der entsprechenden anderen Textur belegt. Hierzu müsste es auch die Funktion geben, wieder auf die Satellitenbilder zu wechseln.

Abbildung 3.12: 3D-Landschaft mit Textur

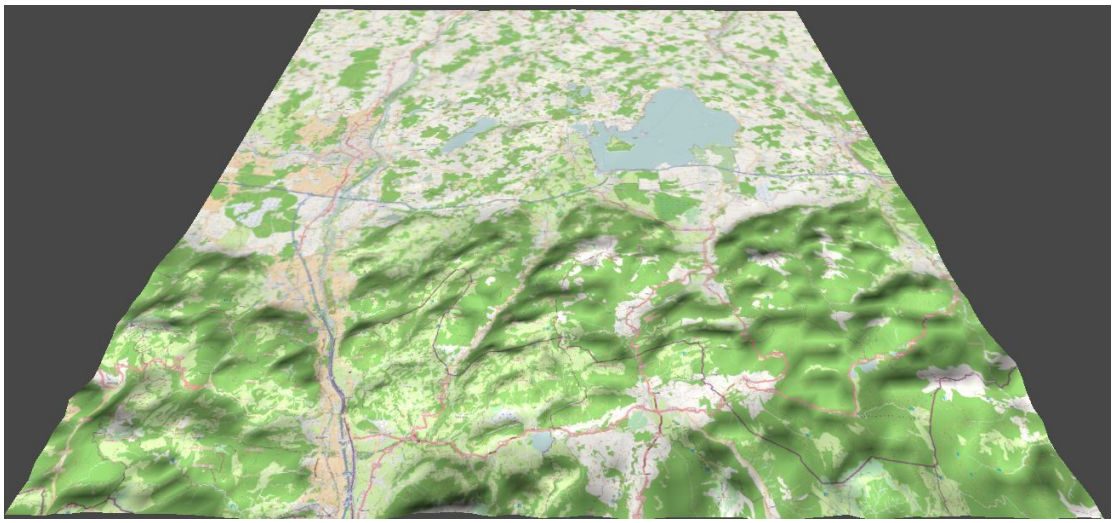


Abbildung 3.13: 3D-Landschaft mit alternativer Textur

Performance und Auflösung

Eine wichtige Frage bei der Darstellung des Terrains ist die der Auflösung. Eine hohe Auflösung der Höhendaten bedeutet, dass Höhenübergänge besser dargestellt werden und auch die Höhenspitzen im Gelände genauer erkennbar sind. Bei den Satelliten bewirkt die höhere Auflösung, dass Details wie z.B. Gebäude und Straßen besser erkennbar sind. Abb. 3.14 stellt zwei verschiedene Auflösungsstufen von ein und der selben Landschaft nebeneinander dar.

Natürlich will man eine gute Auflösung, um eine bestmögliche Darstellung

(a) Hohe Auflösung (b) Niedrige Auflösung

Abbildung 3.14: Vergleich Auflösung

in der Simulation zu erhalten. Zwei Dinge stehen dem allerdings entgegen. Zum einem verringert eine höhere Auflösung die Performance des Simulationsprogramms. Solange der Flug noch flüssig dargestellt wird, ist das kein Problem. Da aber das Programm nicht nur auf Highendrechnern laufen soll, muss hier rechtzeitig reagiert werden. Zum anderen bedeutet eine höhere Auflösung auch einen erhöhten Speicherplatzbedarf. Dies ist vor allem bei der Webplayervariante problematisch. Bei dieser müssen erst alle Daten des Programms vom Server auf den Rechner des Benutzers geladen werden. Dies beinhaltet natürlich auch die Höhen- und Texturdaten.

Um einen guten Kompromiss bei der Auflösung zu erhalten, gibt es mehrere Möglichkeiten. Eine ist die Ausnutzung der Rasterisierung der Terraindaten. Hier kann man die Tatsache nutzen, dass der dargestellte Bereich 450x450 km beträgt, das Gebiet in dem der Hagelabwehrflieger seine Einsätze fliegt, jedoch viel kleiner ist. Da aber hauptsächlich dieses Gebiet von Interesse ist, kann man in diesem Gebiet die Auflösung der Daten höher wählen und im restlichen Gebiet im Gegenzug geringer. Auch sollte man für die Webplayervariante kleinere Auflösungen wählen als für die Standalonevariante, da wie im oberen Abschnitt erwähnt, diese jedesmal über das Internet heruntergeladen werden müssen. Ist die Rasterisierung fein gewählt, kann man schließlich auch zwischen dicht besiedelten Gebieten wie Städten und weniger dicht besiedelten Gebieten wie Bergregionen unterscheiden. Für erstere wählt man eine höhere Auflösung als für die zweite, da hier mehr Informationen vorhanden sind.

Ortsmarkierungen

Zusätzlich zu der normalen Darstellung der Landschaft gibt es noch die Möglichkeit Markierungen anzugeben. Diese werden dann in der Simulation in der Form eines Textes und einer Art Stock, der den zu markierenden Ort berührt, dargestellt (siehe Abb. 3.15). Eingelesen werden die Markierungen mit Hilfe einer xml-Datei. Diese muss denn Namen signs.xml tragen und im selben Ordner liegen wie die auszuführende Datei der Simulation. In dieser werden sowohl der Text, der später darzustellen ist, als auch die Position der Markierung in GPS-Koordinaten, angegeben. Ein Beispiel für die Beschreibung ist in Listing 3.1 zu sehen.

Abbildung 3.15: Ortmarkierung in der Landschaft

3.3.2 Flugsimulation

Bevor man sich Gedanken über die Bewegung des Flugzeuges im Raum macht, braucht man als erstes eine geeignete Repräsentation des Flugzeuges. Für erste Tests mit der Software genügt es, das Flugzeug mit Hilfe eines beliebigen Objektes darzustellen, wie z.B. einer Kugel. Für die fertige Version sollte nach Möglichkeit das Flugzeug möglichst wirklichkeitstreu dargestellt werden. Idealerweise verwendet man ein bereits fertiges 3D-Modell des beim Ha-

```
<Document>
 <Placemark>
    <name>Flugplatz Vogtareuth</name>
    <visibility>0</visibility>
    <Point>
      <coordinates>12.2038888,47.94638</coordinates>
    </Point>
 </Placemark>
</Document>
```

Listing 3.1: Beipiel XML-Marker

gelabwehreinsatz verwendeten Flugzeuges, das dann von Unity nur noch eingelesen und mit wenigen Anpassungen direkt verwendet werden kann. Glücklicherweise konnte ein eben solches Modell von Herrn Prof. Zentgraf gefunden und zur Verfügung gestellt werden. Einzig die Lackierung auf dem Flugzeug unterscheidet sich von der des Hagelabwehrfliegers (siehe Abb. 3.16).

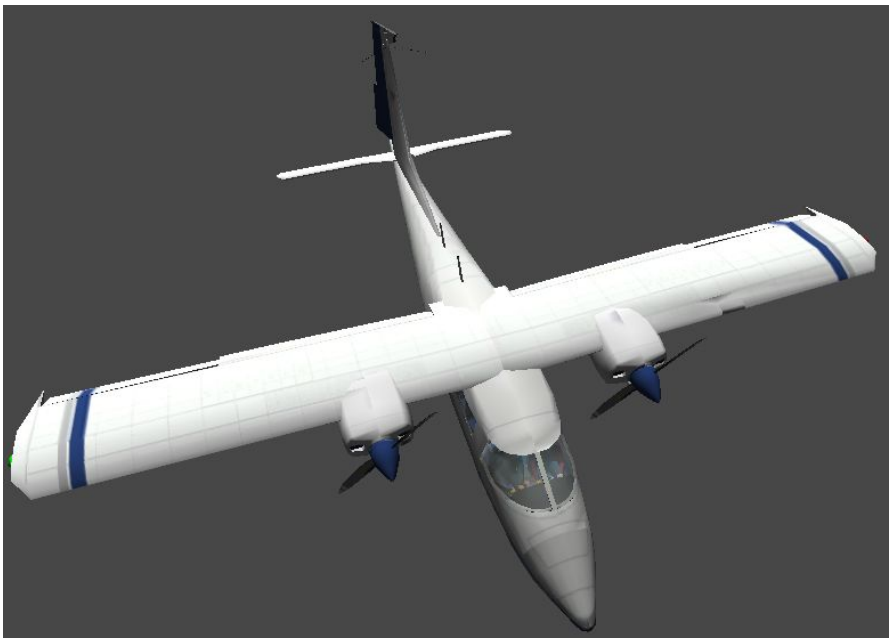


Abbildung 3.16: 3D-Modell Hagelabwehrflieger

Nachdem ein Modell für das Flugzeug vorhanden ist, stellt sich als nächstes die Frage, wie genau denn nun der Flug zu simulieren ist. Als erster Schritt müssen die Positionsdaten und die zugehörigen Zeitangaben so verknüpft werden, dass eine realistische Bewegung des Flugzeuges entsteht. Zusätzlich

wird noch die Lage des Flugzeuges im Raum benötigt. Theoretisch kann man diese aus dem Wissen über die aktuelle Position und die nächste Position errechnen. Dies ist aber recht aufwändig und dank der Tatsache, dass die Informationen von den an Bord des Flugzeuges befindlichen Sensoren gemessen werden, nicht von Nöten. Diese Daten genügen, um einen Flug gut nachbilden zu können.

Flug- und Bodenspur

Eine Flugspur repräsentiert den zwischen dem Start des Flugzeuges und seiner aktuellen Position bereits zurückgelegten Weg. Hier wird dies durch eine einfache Linie im Raum dargestellt. Mit Hilfe der Flugspur kann der Benutzer gut nachverfolgen, welchen Weg das Flugzeug bereits zurückgelegt hat. Auch kann man sich einen Überblick über den Einsatz machen, wenn dieser beendet , ergo das Flugzeug wieder gelandet ist. Die Bodenspur entspricht der Projektion der Flugspur auf die Erdoberfläche. Die Bodenspur ermöglicht es, leicht zu erkennen über welche Orte und Landschaften genau sich das Flugzeug bewegt hat. Mit der Flugspur alleine ist dies schwieriger zu erkennen, da man direkt auf die Szene blicken muss und dadurch die Orientierung eingeschränkt wird. Abb: 3.17 demonstriert die Darstellung von sowohl Luft- als auch Bodenspur. Die Flugspur ist die rot-orange Linie und die bläuliche Linie ist die Bodenspur.

Dargestellt werden sowohl Boden- als auch Flugspur in Unity mit Hilfe der Komponente Line Builder. Diese ermöglicht die Darstellung einer Linie im 3-dimensionalen Raum. Folgende Parameter sind für den Erstellung dieser Linie von Bedeutung:

Datenpunkte: Jene Punkte im 3-dimensionalen Raum, entlang derer die Linie gebildet wird. Es wird entsprechend der Reihenfolge in der Liste der Datenpunkte vorgegangen.

Länge: Die Anzahl der Elemente in der Liste der Datenpunkte, die für das Zeichnen der Linie herangezogen werden sollen. Alle Werte in den Datenpunkten, die nicht von der gewählten Länge eingeschlossen sind, werden bei der Zeichnung der Linie ignoriert.

Anfangs- und Endfarbe: Angabe der Farbe, welche die Linie am Beginn bzw. am Ende haben soll. Dazwischen wird automatisch ein Farbverlauf zwischen den gewählten Farben erzeugt. Durch die Wahl von zwei unterschiedlichen Farben kann man so auf den ersten Blick erkennen, wo der Anfang und wo das Ende der Linie ist.

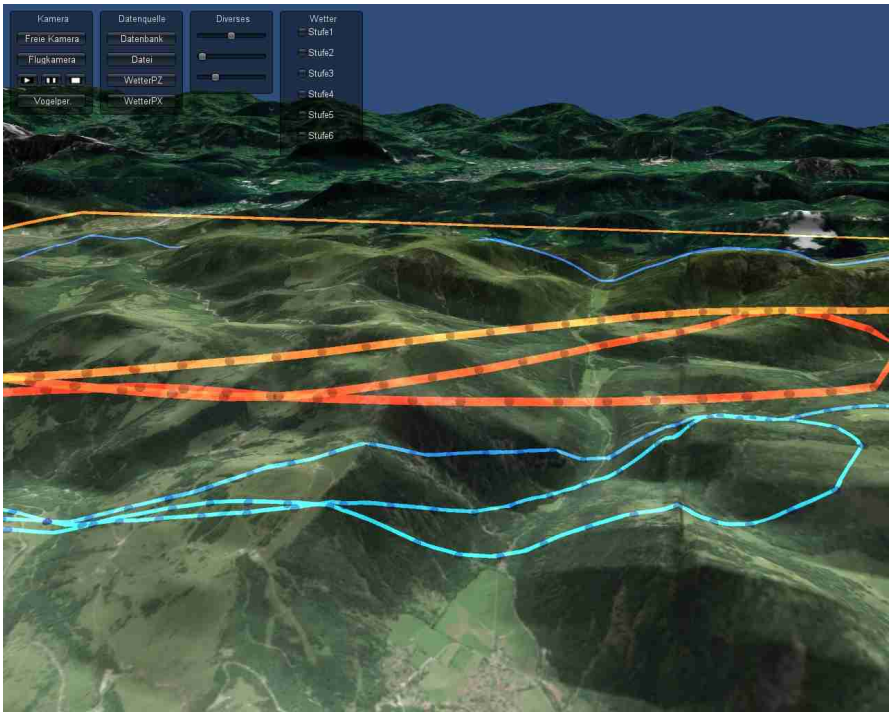


Abbildung 3.17: Flug- und Bodenspur

Die Datenpunkte der Flugspur sind einfach die Positionsdaten des Flugzeugs. Die Datenpunkte der Bodenspur sind bis auf die Höhe mit der Luftspur identisch. Die Höhe wird einfach mit dem Wert für die Höhe des an dieser Stelle liegenden Terrainstücks gleichgesetzt. Damit ist leicht garantiert, dass die Bodenspur auch wirklich am Boden entlang verläuft. Zusätzlich gibt es auch einen Mechanismus, der hauptsächlich dann in Kraft tritt, wenn die eigentlichen Messpunkte des Flugzeuges zu weit auseinander liegen. Wenn man in diesem Fall bei der Bodenspur einfach die Datenpunkte mit den Messpunkten gleich setzt, kann es passieren, dass die Linie durch das Gelände hindurchgeht. Dies ist z.B. dann der Fall, wenn zwischen den fraglichen Punkten eine Erhöhung, z.B. ein Hügel, liegt. Wenn die Punkte nun einfach miteinander verbunden werden, verläuft die Linie unterhalb dieses Hügels und damit auch meist unterhalb der dargestellten Landschaft. Dies hat in letzter Konsequenz eine unschöne optische Unterbrechung der Bodenspur zur Folge. Bei der Flugspur tritt dieses Phänomen normalerweise nicht auf, da das Flugzeug einige hundert Meter über dem Boden fliegt und damit die Verbindungslinie zwischen zwei Punkten nur bei sehr großem Abstand und einer besonders großen Erhöhung zwischen den Punkten optisch unterbrochen wird. Und sollte so etwas trotzdem einmal vorkommen , z.B. bei einer längeren Funkunterbrechung zwischen Flugzeug und Empfangsstation, gewinnt man durch eine Korrektur

dieses Effekts keinen Mehrwert.

Zusätzlich zu der Linie sind die einzelnen Datenpunkte der Bewegung des Flugzeuges noch in Form von Punkten dargestellt. Diese haben in etwa die selbe Farbe wie die dazugehörige Linie, ergo rot für die Flugspur und blau für die Bodenspur. Zwar sind die Positionen dieser Punkte identisch mit den Datenpunkten, aus denen die jeweilige Linie besteht, aber sie sind eigenständige Objekte im 3-dimensionalen Raum und haben ansonsten auch keinen Bezug zu der Flug- bzw. Bodenspur. Im Moment kann man mit Hilfe dieser Punkte hauptsächlich besser sehen, an welchen Stellen es Messdaten vom Flugzeug gibt. Allein anhand der Linie wäre dies nicht zu erkennen, da das Flugzeug auch für eine längere Zeit nur in eine Richtung fliegen kann und dies durch eine gerade Strecke dargestellt werden würde.

Eine weitere vorstellbare Verwendung für die Punkte wäre, sie zur Darstellung der Wetterdaten aus dem Flugzeug zu verwenden. Beispielsweise könnte ein Mausklick auf einen solchen Punkt ein extra Textfeld öffnen, in dem die Wetterinformationen zu dem entsprechenden Zeitpunkt angezeigt werden.

3.3.3 Wettersimulation

Die grafische Repräsentation des Wetters ist ein wichtiger Gesichtspunkt. So sollte es für den Nutzer schnell erschließbar sein, wo gerade ein Hagelzentrum und wie hoch die Niederschlagwahrscheinlichkeit der Wolken ist. Eine erste Überlegung ist, sich möglichst nahe an der Natur zu orientieren. Die einzelnen Messwerte würden dann mit einer Wolkenstruktur dargestellt. Um dann die verschiedenen Intensitäten darzustellen, könnten unterschiedliche Graustufen, ähnlich zu denen echter Wolken verwendet werden. Allerdings stellt sich schnell heraus, dass eine solche Darstellung für die gestellte Aufgabe nicht optimal ist. Zum einem ist bei unterschiedlichen Graustufen und bei großen Wolkengebilden recht schwer die genaue Intensität an einer Stelle zu sehen, auch wenn man eine Farbrefenz der Intensitäten zur Verfügung hat. Zum anderen müsste ein erheblicher Aufwand betrieben werden um eine möglichst realistische Darstellung der Wolken zu ermöglichen. Es gibt zwar einige vorgefertigte Lösungen für Unity, die Wolken schön darstellen können, aber diese sind wiederum nicht kompatibel mit der gegebenen Datenstruktur. Anhand der Datenstruktur selbst Wolken zu modellieren würde dagegen wiederum unnötig viel Zeit in Anspruch nehmen. Daher ist diese Art der Darstellung der Wolken eher ungeeignet.

Alternativ dazu bietet sich eine abstraktere Darstellung der einzelnen Datenwerte an. Statt mit wolkenähnlichen Gebilden wird ein Datum durch ein anderes Objekt dargestellt. Ohne sich näher mit Möglichkeiten zu dieser Darstellung vertraut zu machen, wurde analog zu [Hög09] ein Quader als reprä-

sentatives Objekt gewählt. Um eine größtmögliche Kontinuität zu gewährleisten, wurde ebenfalls die Zuordnung von Farben zu den einzelnen Intensitätsstufen aus [Hög09] übernommen (siehe Abb. 3.18).

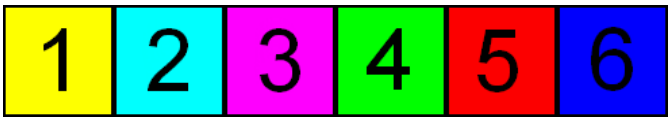


Abbildung 3.18: Wetterintensitäten und ihre Farbzuordnungen

Mit den zugehörigen Daten lässt sich dann das Wetter recht leicht präsentieren. Der Quader als 3D-Objekt ist in Unity bereits als vorgefertigtes Objekt vorhanden. Dieser muss also nur mit einem einzigen Befehl erzeugt, an die richtige Stelle verschoben, passend skaliert und mit der passenden Farbe ausgestattet werden. Das Ergebnis sieht man in Abb. 3.19.

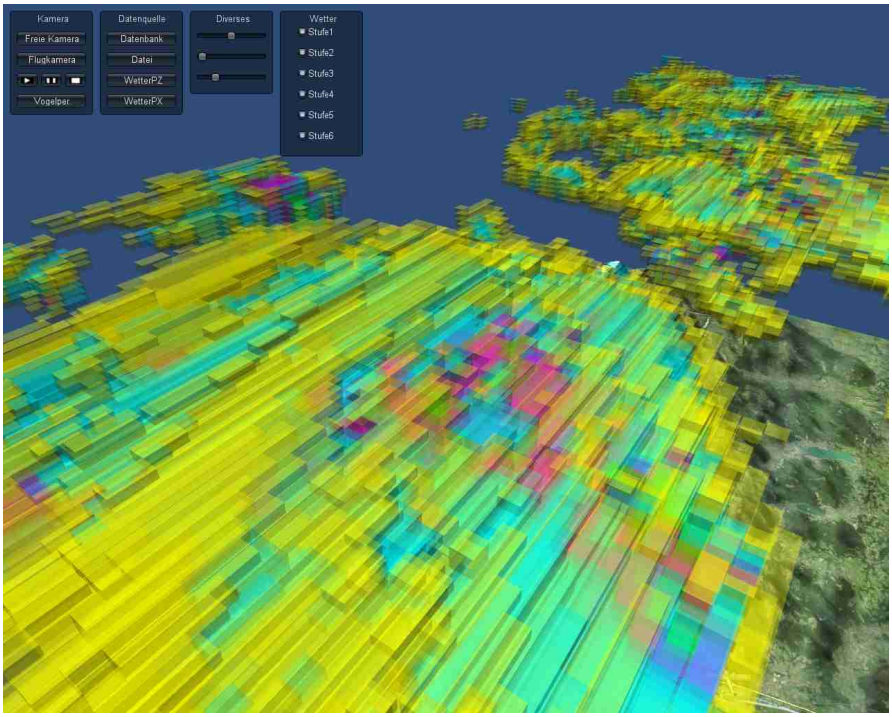


Abbildung 3.19: Beispiel Wetterdarstellung

Überlegungen Performance

Es zeigte sich schnell, dass sich diese Art der Darstellung recht problemlos implementieren lies. Alle Daten würden als Würfel angezeigt und nach ein paar

kleineren Korrekturen auch an der richtigen Stelle in der Simulation dargestellt. Indessen zeigte sich ein anderes Problem. An Zeitpunkten, an denen es stark bewölkt ist, ergeben sich auch viele darzustellende Daten. Dies hat wiederum zur Folge, dass viele einzelne Würfel im Programm angezeigt werden müssen. An diesen Zeitpunkten brach die Performance, also die Bilder pro Sekunde, erheblich zusammen und beim Navigieren durch das Bild kam es zu starken Rucklern. Um also bei vielen anzuzeigenden Daten die Performance trotzdem hochzuhalten, muss ein Konzept überlegt werden, wie die Anzahl der Quader reduziert werden kann, ohne die Art und Weise der Darstellung grundlegend zu ändern. Die grundlegende Idee hierbei ist, die einzelnen Quader „zusammenzukleben". Normalerweise ist zwischen den einzelnen Quadern ein kleiner Abstand. Verbindet man an diesem kleinen Abständen einzelne Quader gleicher Intensität und damit Farbe zu einem größeren Objekt, kann man sich einige Daten, die zur Darstellung nötig sind, sparen. Dazu wird als erstes das Format der Wetterdaten noch einmal genauer betrachtet. Diese sind im Fall von PZ als zwölf und bei PY als ein einzelnes 2-dimensionales Array angeordnet. Mit dieser Information im Hinterkopf werden drei Möglichkeiten der Reduktion der darzustellenden Objekte näher erläutert:

- 1-dimensionale Reduktion
- 2-dimensionale Reduktion mit Rechtecken
- 2-dimensionale Reduktion mit Polygonen

Die 1-dimensionale Reduktion stellt die einfachste Variante dar. Zum besseren Verständnis vergegenwärtigt man sich zuerst einmal einen Ausschnitt des Datenformates (siehe Abb. 3.20)

1	1	1	1	2	2	1	1	1	1	1
1	1	1	2	2	2	1	1	1	1	1
1	1	2	2	3	3	2	2	1	1	1
1	1	1	2	2	3	2	1	1	1	1
1	1	1	1	2	2	2	1	1	1	1
1	1	1	1	1	2	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1

Abbildung 3.20: Ausschnit aus der Datenstruktur Wetter

Wie man sieht, liegen in einer Richtung oftmals viele Werte gleicher Intensität nebeneinander. Dies ist normalerweise bei kleineren Intensitäten wie 1 oder 2 der Fall. Dies spiegelt dann eine echte Unwettersituation wieder, dessen Repräsentation die Daten schließlich auch sind. Bei einer solchen sind die Gebiete leichter und mittlerer Intensität in der Überzahl, es gibt aber nur meist wenige und lokal auch stark begrenzte Zentren starken Niederschlages.

Als erste Reduktion der darzustellenden Informationen fasst man die Daten einer Intensitätsstufe in einer vorher bestimmten Richtung zusammen. Im konkreten Fall heißt dies z.B., dass fünf nebeneinander liegende Werte mit Wert 1, die vorher als 5 einzelne Quader dargestellt wurden, nun als ein einzelner Quader dargestellt werden. Dieser neu entstandene Quader ist dann so groß wie die fünf Einzelquader zusammen. Abb 3.21 veranschaulicht diesen Vorgang anhand der vorher gezeigten beispielhaften Daten.

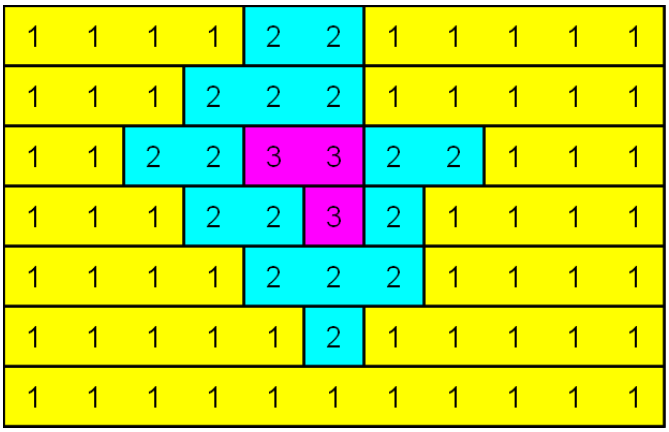


Abbildung 3.21: Beispiel 1-dimensionale Datenreduktion

Ein nächster Schritt der Datenreduktion ist die im 2-dimensionalen mit Hilfe von Rechtecken. Die Grundidee ist die selbe wie bei der 1-dimensionalen Reduktion, aber statt nur in eine Richtung die Daten zusammenfassen, fasst man hier in beide Richtungen zusammen, also 2-dimensional. Man fasst Bereiche gleicher Werte in einem möglichst großen Rechteck zusammen. Ein Rechteck bietet sich hier am besten an, da es ohne große Komplikationen wieder als Quader in Unity dargestellt werden kann. Eine Möglichkeit einer solcher Einteilung wäre mit Hilfe des Quadtree-Verfahren möglich. Bei diesem wird der Anfangsbereich in vier jeweils vier gleich große Teilbereiche unterteilt. Sind alle Werte innerhalb eines solchen Teilbereiches identisch, ist man bei diesem Teilbereich fertig mit dem Algorithmus. Sind die Werte nicht identisch, wird dieser Bereich wiederum in vier gleich große Teilbereiche aufgespalten. Dies geschieht solange, bis man nur noch Bereiche mit gleichen Werten hat. Im Minimalfall enthält ein solcher Bereich nur einen Wert. Anschließend werden

benachbarte Teilbereiche, die gleiche Werte haben, zusammengefasst. Dabei geht man in umgekehrter Reihenfolge vor wie bei der zuvor erfolgten Aufspaltung. Mit diesem Algorithmus kann man Rechtecke mit gleichen Werten gut zusammenfassen. Allerdings ist nicht garantiert, dass dies die minimal mögliche Anzahl an Rechtecken ergibt. Eine Möglichkeit einer solchen Zusammenfassung der Werte sieht man in Abb. 3.22).

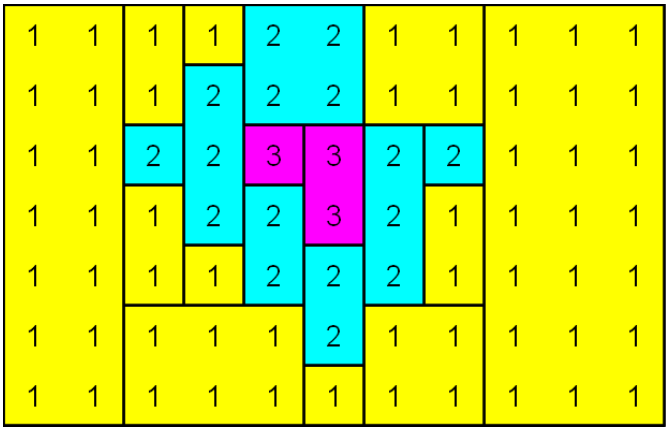


Abbildung 3.22: Beispiel 2-dimensionale Datenreduktion mit Rechtecken

Eine weitere Möglichkeit, die darzustellenden Daten noch weiter zu reduzieren, wäre die Bereiche gleicher Werte so weit wie möglich zusammenzufassen. Solche Bereiche wären dann zwangsläufig nicht mehr einfache Rechtecke, sondern wären dann je nach Zusammensetzung der Daten Polygone mit variabler Anzahl an Ecken. Für das fortlaufende Beispiel würde das wie in Abb. 3.23 aussehen.

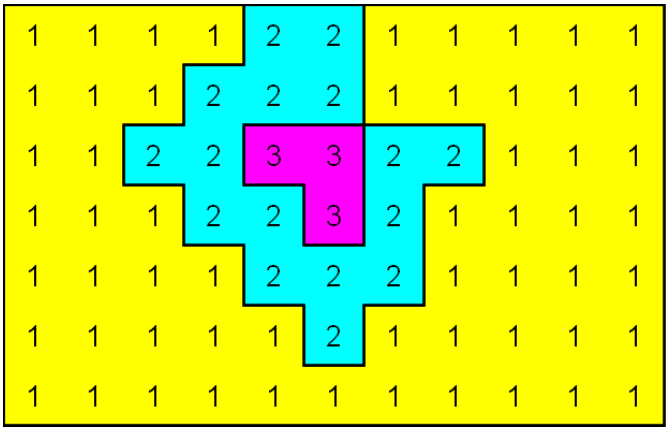


Abbildung 3.23: Beispiel 2-dimensionale Datenreduktion mit Polygonen

Hiermit hat man eine maximale Reduktion der Daten erreicht. Alle Bereiche mit gleichgroßen Werten, die an einer beliebigen Stelle Berührungspunkte haben, würden als ein einziges Objekt dargestellt. Die praktische Umsetzung dieser Idee erweist sich allerdings als sehr schwierig. Ein Algorithmus müsste als erstes diese Bereiche finden und anschließend daraus die nötigen Informationen gewinnen, um ein Polygon zu definieren. Einen solchen Algorithmus zu erstellen wäre nicht ein großes Problem. Allerdings die Umsetzung eines solches Polygons in Unity ist nicht ohne viel Aufwand möglich. Wurde bei den vorherigen Ideen lediglich das leicht zu generierende Quaderobjekt entsprechend skaliert und positioniert, so muss hier ein gänzlich neues 3D-Objekt generiert werden. Im schlimmsten Fall ist es sogar nötig, sämtliche Parameter, die zur Darstellung des 3D-Objekts notwendig wären, selbst zu definieren. Ein solcher Aufwand wäre nur dann zu rechtfertigen, falls die Performance wirklich benötigt wird und keine anderen Alternativen wie z.B. Verringerung des Bereiches der Darstellung, mehr möglich sind. Am Schluss ist noch anzumerken, dass für die PZ-Daten theoretisch auch ein Zusammenfassen der Würfel in der dritten Dimension, also der Höhe, denkbar wäre.

Für die Simulation wurde die Reduktion der darzustellenden Würfel mit Hilfe des Zusammenfassen der Werte im 1-dimensionalen umgesetzt. Die Implementierung war schnell und ohne Probleme erledigt und führt bereits zu einer guten Verbesserung der Performance.

Positionierung

Wie bereits vorher erwähnt, sind die Positionen der Wetterdaten aus einer Karte mit einer anderen Projektionsart erhoben worden als der Gauß-Krüger-Projektion, nämlich einer polarstereographischen Projektion. Um die Wetterdaten in die Simulation einbinden zu können, benötigt man ein paar Eckdaten. Vom DWD selbst stammen die GPS-Koordinaten der vier Punkte, die jeweils das in der entsprechenden Projektion rechteckige Gebiet der PX- und PZ-Daten aufspannen. Diese Punkte sind in den Tabellen 3.4 und 3.5 zu nachzulesen.

Die Wetterdaten können mit Hilfe dieser Eckpunkte richtig im Unity-Koordinatensystem positioniert werden.

3.4 Bedienungskonzept und GUI

Die vorherigen Kapitel haben allesamt die Verarbeitung von Daten und deren Präsentation beschrieben. Ebenfalls wichtig ist aber auch die Schnittstelle zwischen Programm und Benutzer. Diese sollte eine möglichst eingängige Steuerung erlauben.

Links oben geo. Breite: 49,1916° N geo. Länge: 10,3467° E	**Rechts oben** geo. Breite: 49,145° N geo. Länge: 12,9384° E
Links unten geo. Breite: 47,5067° N geo. Länge: 10,3333° E	**Rechts unten** geo. Breite: 47,46° N geo. Länge: 12,81° E

Tabelle 3.4: Eckpunkte PX-Daten

Links oben geo. Breite: 50,0433° N geo. Länge: 9,0367° E	**Rechts oben** geo. Breite: 49,945° N geo. Länge: 14,33° E
Links unten geo. Breite: 46,67° N geo. Länge: 9,1167° E	**Rechts unten** geo. Breite: 46,5783° N geo. Länge: 13,9667° E

Tabelle 3.5: Eckpunkte PZ-Daten

Freie Kamera

Dies ist die Standardeinstellung des Programmes. Mit der freien Kamera ist eine völlig freie Bewegung im Raum möglich. Das heißt, man kann das Geschehen aus jedem gewünschten Blickwinkel heraus betrachten. Dies kann allerdings auch problematisch sein. Dann nämlich, wenn man so sehr mit der Kamera herum experimentiert hat, dass man wesentliche Details, in diesem Fall das Flugzeug, gar nicht mehr im Blick hat. Hier ist eine Funktion hilfreich, welche die Kamera wieder in Sichtweite des Flugzeuges bringt und auf dieses zentriert. Es ist aber auch gewünscht, dass man einen Überblick aus großer Höhe hat, um so das gesamte Gebiet mitsamt den Wetterdaten zu überblicken. Auch hier bietet sich eine Funktion an, die dies für einen schnell erledigt ohne dafür lange mit der Kamera navigieren zu müssen.

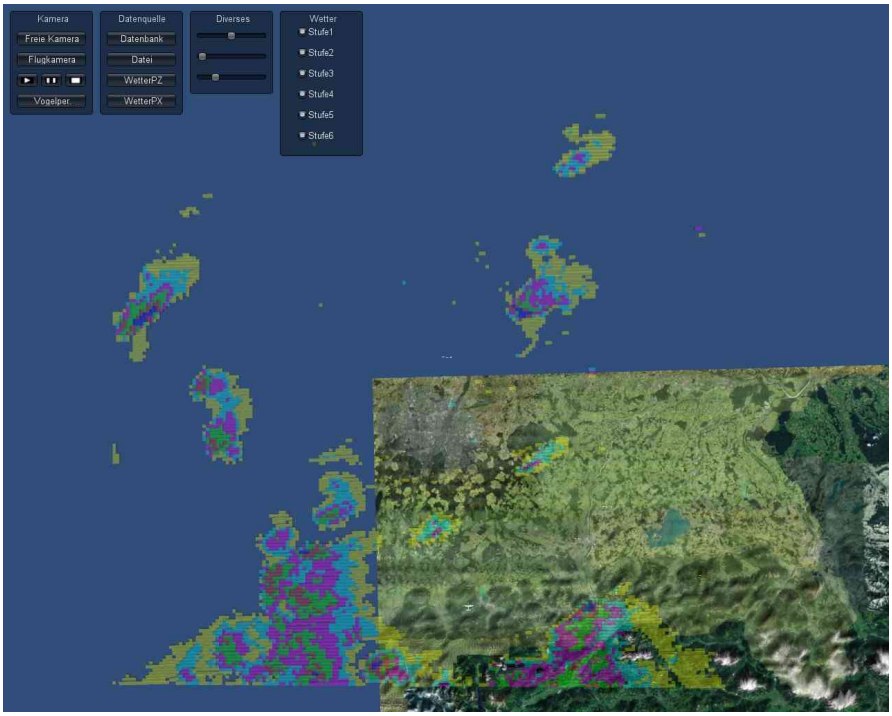


Abbildung 3.24: Ansicht freie Kamera

Cockpitkamera

Die Cockpitkamera ist an das Flugzeug gebunden. Man kann sich zwar frei umschauen, die Position der Kamera ist aber an die Position des Flugzeuges gekoppelt. Daher bewegt sich diese Kamera nur dann, wenn sich auch das Flugzeug bewegt. Damit ist man im Gegensatz zur freien Kamera immer nahe an dem aktuellen Einsatzgebiet des Hagelabwehrfliegers. Durch die Koppelung an das Flugzeug ist es unmöglich, dass man die Orientierung verliert. Auch hat man durch das Umschalten auf die Cockpitkamera die Möglichkeit, wieder das Geschehen zu verfolgen, falls man mit der freien Kamera die Szene aus den Augen verloren hat.

Navigationselemente

Die Navigation innerhalb der Simulation ist von enormer Bedeutung. Diese sollte leicht von der Hand gehen, damit sich der Nutzer auf das Geschehen innerhalb der Simulation konzentrieren kann. Die momentan verwendete Steuerung der Kamera sieht diese als direkt bewegbares Objekt im 3-dimensionalen Raum vor. Über die Pfeiltasten kann die Kamera in die entsprechende Richtung bewegt werden. Alternativ dazu können auch die Tasten W, A, S und D

Abbildung 3.25: Ansicht Cockpitkamera

analog zu den Pfeiltasten oben, links, unten bzw. rechts verwendet werden. Um sich mit der Kamera umzuschauen, ohne dass sich die Kamera bewegt, wird die rechte Maustaste gedrückt gehalten. Danach kann man durch Bewegung der Maus die Blickrichtung der Kamera entsprechend der Mausbewegung ändern. Dies ist nur solange möglich, wie die rechte Maustaste gedrückt bleibt. Dieser sogenannte „Mausblick“ kann problemlos mit der normalen Bewegung der Kamera kombiniert werden. Diese beiden Steuerungselemente reichen aus, um eine gute Navigation innerhalb der Simulation zu ermöglichen.

Eine weitere Möglichkeit wäre, sich an der Steuerung von Google Earth zu orientieren. Dies hätte den Vorteil, dass viele Leute damit vertraut sind. Da Google Earth ebenfalls eine Simulation der Erde beinhaltet, würde sich eine ähnliche Steuerung bei diesen artähnlichen Programmen durchaus anbieten. Da allerdings die bisherige Steuerung ohne große Probleme funktioniert, wird das mögliche Implementieren dieser Art der Steuerung auf die Zeit nach dem Abschluss dieser Arbeit verschoben.

Steuerung Flugzeugbewegung

Nach dem Laden der Positionsdaten des Flugzeuges gibt es genau drei Steuerungselemente, Abb. 3.26 zeigt die graphische Darstellung dieser Elemente. Die Funktionalitäten dieser Steuerungselemente sind im Einzelnen:

(a) Abspielen (b) Pause (c) Stopp

Abbildung 3.26: Tasten Flugzeugsteuerung

Abspielen: Startet den Flug. Wenn der Flug pausiert wurde, dann setzt ein Druck auf diese Taste den Flug fort. In allen anderen Fällen passiert nichts beim Betätigen von „Abspielen“.

Pause: Stoppt den Flug an der aktuellen Stelle. Sinnvoller Einsatz nur dann, wenn gerade ein Flug in Gange ist. Sonst passiert bei Druck dieser Taste nichts.

Stopp: Bricht den aktuellen Flug ab. Flug- und Bodenspur werden entfernt und das Flugzeug an seinen Ausgangspunkt zurückgesetzt. Die Kamera wird auf die Position des Flugzeuges zentriert. Die geladenen Positionsdaten bleiben vorhanden. Mit dem Betätigen von „Stopp“ und anschließend „Abspielen“ wird der Flug effektiv neu gestartet.

Bedienelemente Wetter

Wie in den vorherigen Kapiteln erläutert, besteht die Wetterdarstellung aus sechs verschiedenfarbigen Quadern. Um sich einen besseren Überblick über die Wettersituation verschaffen zu können, gibt es die Möglichkeit die Darstellung einzelner Wetterintensitäten auszuschalten. Dies ist besonders nützlich für die Deaktivierung der Darstellung der niedrigen Wetterintensitäten wie z.B. Stufe 1, 2, 3. Besonders bei Wetterlagen mit viel Niederschlag gibt es von niedrigen Stufen sehr viele darzustellende Elemente. Durch das Ausblenden dieser Intensitäten sieht man nicht nur die Unwetterzentren besser, d.h. die höheren Niederschlagsintensitäten, sondern auch die Darstellung ist dadurch performanter, da weniger angezeigt werden muss.

Kapitel 4

Ergebnis/Beispiel

4.1 Ausgangssituation

Als Beispiel für die Funktionsweise der Software soll ein realer Hagelabwehreinsatz dienen. Ein solcher fand am 02.08.2012 statt. Folgende Daten sind für diesen Flug verfügbar:

Daten Flugzeug: Die für spätere Einsätze vorgesehenen Sensoren (HAILsens, HAILaquisition) waren noch nicht an Bord. Auch die geplante Funkverbindung war noch nicht einsatzbereit. Stattdessen war ein von den anderen Komponenten unabhängiger GPS-Empfänger im Flugzeug. Dieser sendete seine Daten an einen Empfänger an der Fachhochschule. Folglich beinhalteten die Daten diese Informationen: geographische Breite, geographische Länge, Höhe und die Uhrzeit. Mit diesen Daten kann man einen Einsatz des Hagelabwehrflugzeuges bereits nachstellen. Zu bemerken ist noch, dass der zeitliche Abstand zwischen den gemessenen Daten bei 5 Sekunden liegt. Für den Produktiveinsatz liegt der zeitliche Versatz bei 1 Sekunde, auf die Darstellung wirkt sich dieser Unterschied allerdings nicht merklich aus. Auf dem Computer liegen die Daten in Form einer simplen .txt-Datei vor. Eine Zeile aus dieser Datei ist in Listing 4.1 zu sehen.

```
MOBILE>BASE: $GPGGA,170019.000,4757.0671,N,
01213.5828,E,1,10,1.0,618.9,M,47.2,M,,0000*52
```

Listing 4.1: Auszug Flugdaten

Die Zeilenunterbrechung dient hier nur der besseren Lesbarkeit, in der Datei selber entspricht dies genau einer Zeile und damit einem Messpunkt. Wie man gut erkennen kann, stehen die Werte für geographische Länge und Breite jeweils vor ihrem jeweiligen Bezeichner für die geographische Orientierung, in diesem Beispiel also N bzw. E. Ungewöhnlich ist allerdings die Art der Notation. Am gebräuchlichsten ist bei GPS-Daten die Angabe in Dezimalgrad, z.B. 48,1204°. Hier ist die Notation

dagegen erst Grad und dann Dezimalminuten, was für genanntes Beispiel folgende Schreibweise bedeutet: 48° 7,224'. Bei der Breitenangabe sind die ersten beiden Ziffern die Gradangabe, der Rest die Dezimalminuten. Zwei Ziffern genügen hier für die Gradangabe, da der Wert nicht größer als 90° sein kann. Bei der Angabe der Länge sind die ersten drei Ziffern für die Gradangabe reserviert, der Rest sind wieder die Dezimalminuten. Die Zeit in Form der UTC-Zeit ist die erste Zahl in der Zeile und hat das Format: hhmmss.msmsms. Die Angabe der Höhe erfolgt in Meter und ist die letzte Zahl vor M.

Daten Wetter: Für den Zeitraum des Hagelabwehreinsatz am 02.08.2012 sind sämtliche Wetterdaten vorhanden. Sowohl die PX- als auch die PZ-Daten sind vom DWD bereits zur Verfügung gestellt worden. Auch in die interne Datenbank sind die Werte bereits übertragen worden. Damit liegen die Wetterdaten in Form und Umfang bereits so vor, wie für die fertige Applikation vorgesehen.

4.2 Ablauf

Nachdem die Simulation gestartet ist, wird als Erstes die Datei mit den Positionsdaten des Fliegers geladen. Da diese Daten nicht in der später vorgesehenen Form vorhandenen sind, erfolgt das Einlesen über einen provisorischen Algorithmus, der im fertigen Programm keine Verwendung mehr findet. Dazu werden mit Hilfe von regulären Ausdrücken die benötigten Daten aus der Textdatei herausgesucht und schließlich weiter verarbeitet.

Die Wetterdaten des DWD wurden für dieses Beispiel das erste mal aus einer Datenbank ausgelesen. Für dieses Beispiel wird die nötige Datenbank und der Webserver mit Hilfe einer xampp-Umgebung lokal zur Verfügung gestellt (siehe [XAM]). In allen vorherigen Testversionen des Programmes wurden die Wetterdaten noch statisch aus einer einzelnen csv.-Datei ausgelesen. Dieses statische Auslesen wäre für die Präsentation eines ganzen Hagelabwehrfluges äußert ungünstig, da mehrere Wetterdaten zu den einzelnen Zeitpunkten während des Hagelabwehrfluges benötigt werden. Über die bisher vorhandene GUI kann aber jeweils nur eine Datei mit Wetterdaten ausgelesen werden. Dies wäre deshalb nur unnötig kompliziert. Da bei diesem Testbeispiel die Eingangsparameter für die Wetterdaten statisch sind, ist auch das Auslesen der Wetterdaten aus der Datenbank statisch. Die Wetterdaten werden zusammen mit ihren zugehörigen Informationen zum Aufnahmezeitpunkt der Daten ausgelesen und entsprechend zugeordnet. Dies ist deshalb nötig, damit die Wetterdarstellung während der Simulation des Fluges auch zu den richtigen Zeitpunkten umgeschaltet werden kann.

Im Folgenden werden markante Stellen während des Hagelabwehrfluges, wie sie in der Simulation zu sehen sind, in Form von Bildern dargestellt. Damit gewinnt man nicht nur einen Eindruck, wie die Simulation in ihrer endgültigen Fassung einmal aussehen wird, sondern auch einen kurzen Einblick über die Einsatzweise eines Hagelabwehrfliegers.

Abb. 4.1 stellt den Start des Hagelabwehrfluges dar. Das Flugzeug startet gerade vom Flugplatz Vogtareuth. Im Hintergrund ist bereits das Unwetterzentrum zu erkennen, dargestellt durch die Ansammlung vieler roter Würfel.

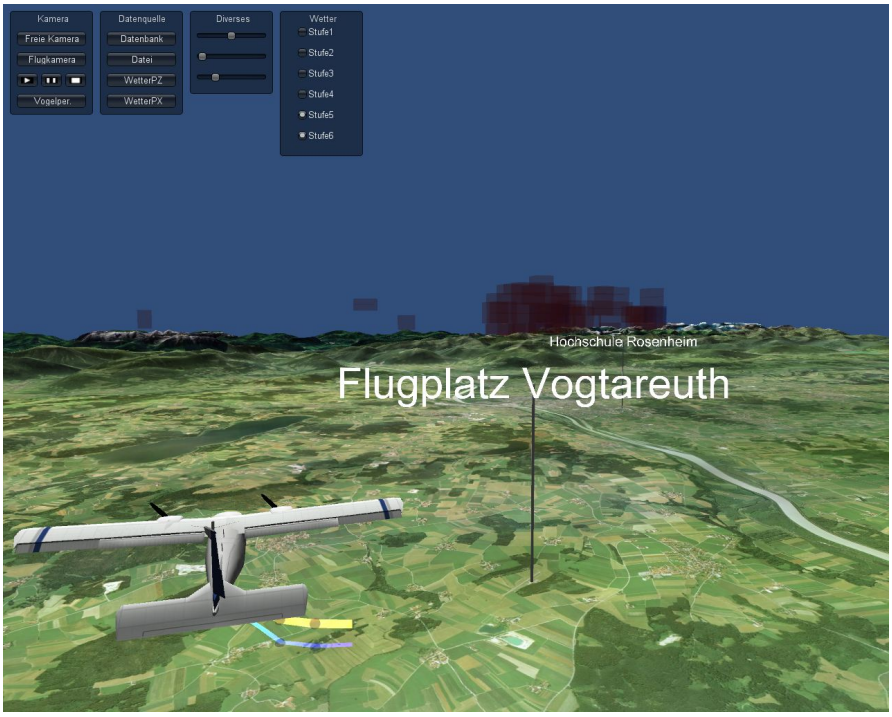


Abbildung 4.1: Wettersituation 17:00 Uhr

Bei Abb. 4.2 um ca. 17:42 Uhr ist der Hagelabwehrflieger bereits mitten im Zentrum des Unwetters. Gut zu sehen ist das Unwetterzentrum durch die blauen Würfel, die für höchste Niederschlagsintensität stehen, und die vielen roten Würfel drum herum.

In Abb. 4.3 um ca. 18:12 Uhr ist das Unwetterzentrum bereits ein Stück nach Osten gewandert. Schön ist hier auch zu sehen, dass das Flugzeug am Ort des Unwetterzentrums mehrere Schleifen fliegt, um eine möglichst große Menge an Silberiodid an dieser Stelle verteilen zu können.

Abb. 4.4 zeigt das Ende des Hagelabwehreinsatzes. Der Flieger befindet sich im Landeanflug. Das Unwetterzentrum ist im Vergleich zu Abb. 4.3 kleiner geworden.

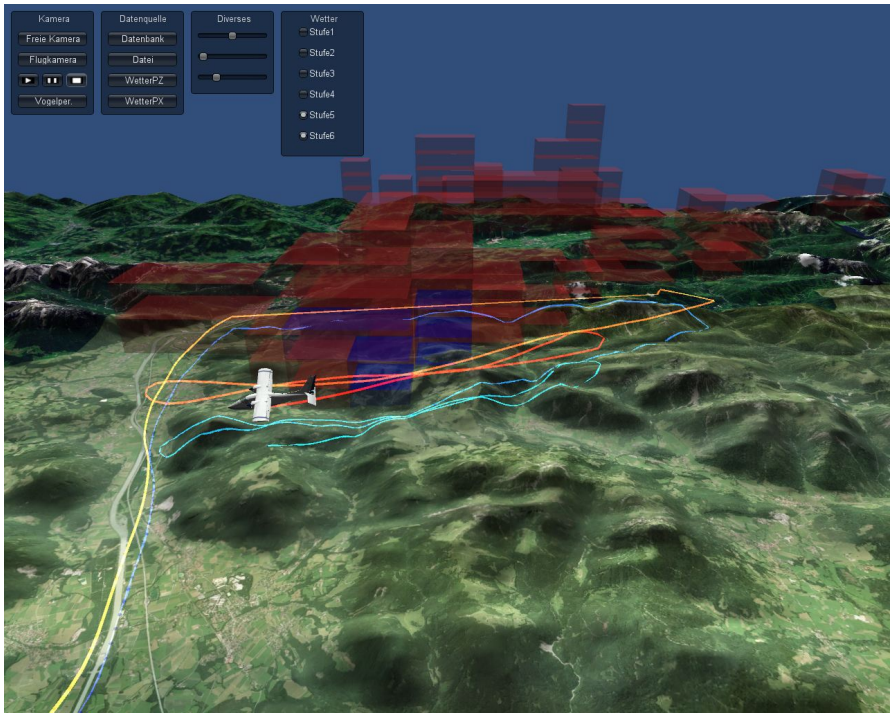


Abbildung 4.2: Wettersituation 17:42 Uhr

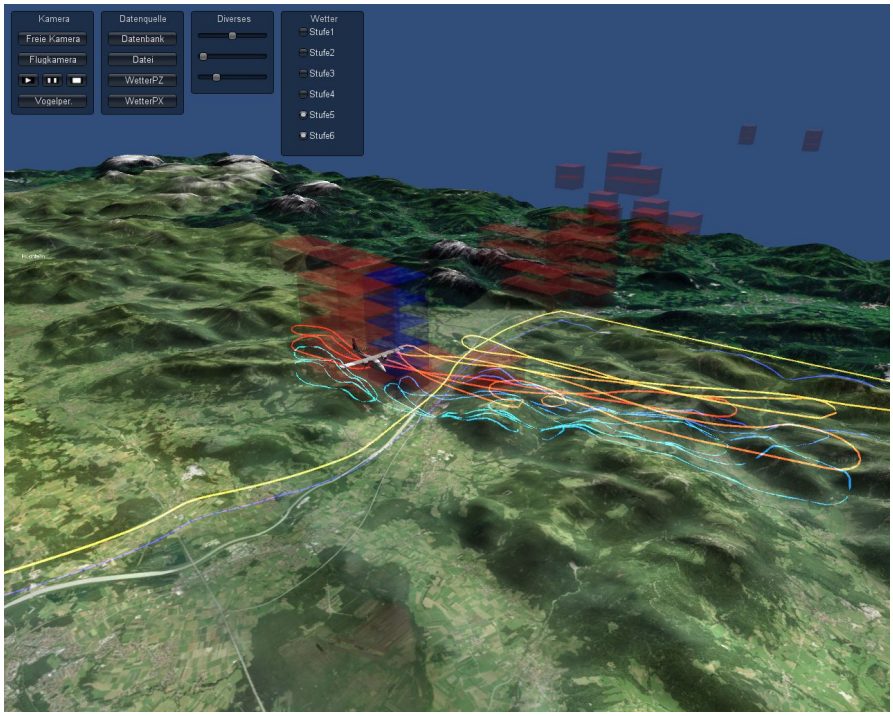


Abbildung 4.3: Wettersituation 18:12 Uhr

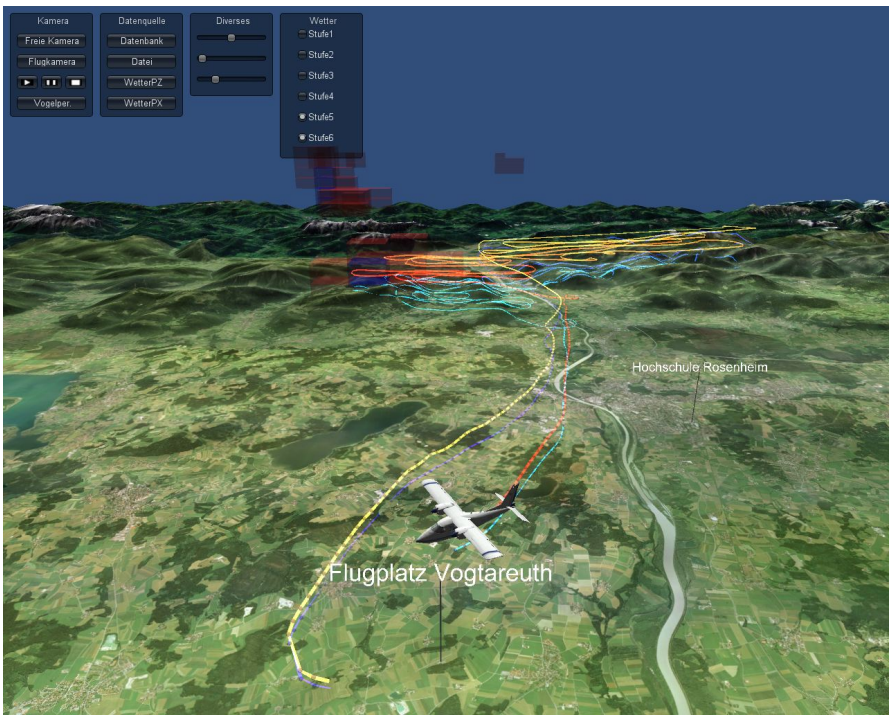


Abbildung 4.4: Wettersituation 18:38 Uhr

Kapitel 5

Fazit

5.1 Aktueller Stand

Das wichtigste Ziel dieser Arbeit, nämlich ein gutes Grundgerüst für die Simulation von Hagelabwehrflügen zu schaffen, wurde erreicht. Auch gibt es eine bereits funktionierende Version der Simulation, die in Kaptiel 4 näher beschrieben wurde. Anhand der Eingangsdaten soll daher zusammengefasst werden, was bereits funktioniert bzw. implementiert und was theoretisch angedacht, aber noch nicht in der Simulation vorhanden ist:

Terraindaten: Die Darstellung des Terrains mit seinem Höhenprofil und den Satellitenbildern der Landschaft funktioniert problemlos. Allerdings wird nicht der ganze Bereich, wie in Kapitel 3.2.1 beschrieben, sondern nur ein kleinerer Bereich, der im Großen und Ganzen dem üblichen Einsatzbereich des Hagelfliegers entspricht. Der nächste Schritt zur Darstellung der ganzen Landschaft ist unproblematisch, da die Gewinnung der Daten und die Implementierung dieser in der Simulation identisch ist zu dem bisher verwendeten Datensatz.

Wetterdaten: Die Anzeige der Wetterdaten funktioniert auch ohne Probleme. Form und Art der Darstellung ist auch final. Ebenso funktioniert das partielle Ausblenden von einzelnen Intensitätsstufen tadellos. Nur die Performance ist in der Darstellung von allen Intensitäten bei starken Unwettern noch nicht optimal. Da aber die Intensitätsstufen ausgeblendet werden können und hauptsächlich die hohen Intensitäten von Bedeutung sind, ist dies ein geringeres Problem.

Flugzeugdaten: Die Bewegung des Flugzeuges in der Simulation anhand der GPS-Daten ist soweit implementiert. Auch die Flug- und Bodenspur wird korrekt mit Hilfe dieser Daten dargestellt. Die Wetterdaten aus dem Flugzeug haben aber noch keinen Platz in der Simulation. Im Gegensatz zu den GPS-Daten benötigen diese allerdings keine weitere Verarbeitung. Sie müssen deshalb einfach nur ausgelesen werden und ihr Wert kann

in der Simulation dargestellt werden, in welcher Weise auch immer. Das Auslesen der Flugdaten erfolgt im Moment noch aus Textdateien. Das Auslesen dieser Daten aus der Datenbank sollte keine große Schwierigkeit darstellen. Eine erfolgreiche Verbindung zur Datenbank wurde bereits mit dem Auslesen der DWD-Wetterdaten hergestellt. Die Positionsdaten des Fliegers können dann auch einfach aus der Datenbank ausgelesen und nach der nötigen Koordinatentransformation für die Darstellung der Flugzeugbewegung genutzt werden.

Der aktuelle Stand der Simulation ist nicht in der Lage, dynamisch einen beliebigen Hagelabwehreinsatz darzustellen. Dass dies theoretisch mit dem bereits erstellten Teil der Software möglich ist, beweist das Beispiel aus Kapitel 4. Trotzdem ist die bisherige Version der Simulation für die Beteiligten an RO-BERTA brauchbar. Flüge, die in Form einer wie in Kapitel 4 beschriebenen Textdatei vorliegen, können abgespielt werden. Auch einzelne Wetterdaten, sowohl PX- als auch PZ-Daten, können dargestellt werden, wenn diese als csv-Datei verfügbar sind.

5.2 Ausblick

Auch wenn die Simulation im Wesentlichen bereits funktioniert, ist noch einiges zu erledigen, bevor die Arbeit als abgeschlossen bezeichnet werden kann. Neben den einzelnen Punkten, die im vorherigen Abschnitt aufgezählt wurden, ist die Anpassung der Webvariante von besonderer Bedeutung. Die bisherige Version, die der Standalone-Variante entspricht, muss nur noch anhand der entsprechenden Vorgaben erweitert und angepasst werden. Die Webvariante kann aus der selben Vorlage wie Standalone-Variante erstellt werden, da die entsprechenden Beschränkungen bereits berücksichtigt wurden. Allerdings muss sie zusätzlich auf einer Webseite für alle erreichbar sein. Ebenfalls darf die Simulation keine übermäßige Dateigröße haben, damit auch Personen mit langsameren Internetanschluss das Programm nutzen können.

Die Anpassungen am Programm und das Integrieren in die bestehende Infrastruktur werden auch nach Abschluss dieser Arbeit (Anfang Januar 2013) vom Autor derselbigen weitergeführt. Das gesamte RO-BERTA-Projekt und damit auch dieses Teilprojekt laufen bis April 2013. Etwa zu dieser Zeit beginnt auch die Saison für die Hagelabwehrflüge. Deshalb wäre es optimal, wenn die Simulation bis dahin einwandfrei funktioniert und alle in dieser Arbeit beschriebenen Funktionen erfüllt. Wenn die für die Simulation wichtigen Systeme, also insbesondere die Datenbanken, bis zu diesem Termin fertig sein sollten, dürfte auch der Fertigstellung der Simulation für die Hagelabwehrflüge nichts im Weg stehen.

Auch wenn die Simulation im April 2013 einwandfrei funktionieren sollte, ist dies noch nicht das Ende aller Möglichkeiten. Zusätzliche Optionen könnten in die Simulation eingebaut werden, um den Mehrwert für Laien und Experten zu erhöhen. Eine besonderes interessante Option wäre das Einbinden von Daten, die von Mitgliedern des Hagelforschungsverein gesammelt und dem Hagelforschungsverein zur Verfügung gestellt werden. Diese bestehen hauptsächlich aus Informationen über die Auswirkungen des Hagels mitsamt Fotos. Vorstellbar wäre es, diese bei Abspielen von alten Einsätzen am Ort der Meldung in der Simulation aufrufbar zu machen. Dies würde das Abschätzen von der Wirksamkeit der Hagelabwehreinsätze noch leichter machen.

Literaturverzeichnis

[Ben06] M. Bender und M. Brill. *Computergrafik*. Hanser, 2006.

[Ber12a] A. Bernhardt. *Entwicklung eines Microcontroller-basierten Prototypen für die echtzeitfähige Sensordatenverarbeitung im Rahmen des Hagelflieger-Projekts RO-BERTA*. Diplomarbeit, Fachhochschule Rosenheim, 2012.

[Ber12b] A. Bernhardt, M. Heigl, P. Viehhauser und P. Zentgraf. Zwischenbericht Projekt RO-BERTA Oktober 2011 - März 2012. Fachhochschule Rosenheim, 2012.

[DWD] Wetterradar in Deutschland. http://www.deutscher-wetterdienst.de/lexikon/download.php?file=Radarverbund.pdf. Zuletzt aufgerufen: 12 Dez 2012.

[Eur] Map Projections for Europe. http://www.crs-geo.eu/crseu/EN/References/Elemente/pub04MapProjectionForEurope.pdf. Zuletzt aufgerufen: 12 Dez 2012.

[Fre11] M. Frey. *Integration einer Sensorik zur Messung von Umweltparametern in eine Flugzeugnase*. Diplomarbeit, Fachhochschule Rosenheim, 2011.

[Glo] Homepage Global Mapper. http://www.globalmapper.com/. Zuletzt aufgerufen: 12 Dez 2012.

[Goo] deutsche Homepage von Google Earth. http://www.google.de/earth/index.html. Zuletzt aufgerufen: 12 Dez 2012.

[Hag] Homepage Hagelabwehrverein Rosenheim. http://www.hagelabwehr-rosenheim.de/. Zuletzt aufgerufen: 12 Dez 2012.

[Hög09] T. Höglauer. *Aufbereitung und Online-Visualisierung der Messdaten eines Hagelabwehrflugs*. Diplomarbeit, Fachhochschule Rosenheim, 2009.

[Kar] Kartierungsstelle des AHO Baden-Württemberg. http://www.orchids.de/florkart/frameset.htm. Zuletzt aufgerufen: 12 Dez 2012.

[L3D] Homepage L3DT. http://www.bundysoft.com/L3DT/. Zuletzt aufgerufen: 12 Dez 2012.

[Mon] MySQL-Schnittestelle für Mono. http://www.mono-project.com/MySQL. Zuletzt aufgerufen: 12 Dez 2012.

[MYS] deutsche mySQL Homepage. http://www.mysql.de/. Zuletzt aufgerufen: 12 Dez 2012.

[ROB] Infoseite zu RO-BERTA. http://www.fh-rosenheim.de/ro-berta.html. Zuletzt aufgerufen: 12 Dez 2012.

[Uni] Unity Homepage. http://unity3d.com/. Zuletzt aufgerufen: 12 Dez 2012.

[WGS] Definition WGS 84. http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf. Zuletzt aufgerufen: 12 Dez 2012.

[XAM] XAMPP Homepage. http://www.apachefriends.org/de/xampp.html. Zuletzt aufgerufen: 12 Dez 2012.