Technische Universität Braunschweig | Gauß-IT-Zentrum
Hans-Sommer-Straße 65 | 38106 Braunschweig | Deutschland

VESPER Abschlussbericht Anlage A1

Dokument:	Wissenschaftlich-technische Ergebnisse
Verbundprojekt:	Verbesserung der Sicherheit von Personen in der Fährschifffahrt (VESPER)
Teilvorhaben:	Informationsvisualisierung in der Mensch-Maschine-Schnittstelle
FKZ:	13N9653

Zuwendungsempfänger:	Technische Universität Carolo-Wilhelmina zu Braunschweig – Gauß-IT-Zentrum
Projektleiter:	Carsten Hellmich
Laufzeit des Vorhabens:	01.03.2009 – 31.05.2011
Anlagen:	

Inhalt

1 Virtueller Fährhafen

Für die Arbeitspakete 2.2.2 und 2.2.3 entstand eine Software zur Visualisierung und Simulation der Passagier- und Fahrzeugkontrollen in einem virtuellen Fährhafen. Im Rahmen des Gesamtvorhabens wurde vom Fraunhofer FKIE ein Prozessmodell für die Abfertigung der Passagiere erstellt, das als Grundlage zur simulierten des Objektverhaltens verwendet wurde.

Abb. 1: Szenario im virtuellen Fährhafen

1.1 Anforderungen

Folgende Funktionen wurden identifiziert:

- Es wird eine realistische 3D-Hafenumgebung mit frei beweglicher Kamera benötigt.
- Auf dem virtuellen Gelände befinden sich Personen.
- Personen können Passagiere und Hafenmitarbeiter sein.
- Auf dem Gelände existieren Straßen, die von Fahrzeugen befahren werden.
- Fahrzeuge können Passagiere aufnehmen.
- Fahrzeuge können mit der Fähre reisen oder zur Beladung dienen.
- Passagiere (mit und ohne Fahrzeug) verhalten sich „intelligent“.
- Passagiere ohne Fahrzeug werden von einem Shuttle-Bus transportiert.
- Die Fähre legt regelmäßig an und ab.
- Personen und Fahrzeuge können die Fähre betreten und verlassen.
- Sicherheitssysteme können zur Laufzeit platziert werden.
- Personen und Fahrzeuge werden durch die Sicherheitssysteme auf gefährdende Aktionen und Gegenstände überprüft.
- Die Funktionsweise der Sicherheitssysteme wird visualisiert

- Es können automatisierte Ereignisse ablaufen (Szenarien)
- Die Simulationsgeschwindigkeit kann interaktiv verändert werden
- Während der Simulation werden Daten aufgezeichnet, die eine Effizienzbewertung der eingesetzten Sicherheitstechnologien ermöglicht

Folgende technische Vorgänge wurden identifiziert:

- Die virtuelle Hafenumgebung wird mit einem Editor erstellt und gespeichert.
- Die Objekttypen und deren Eigenschaften werden mit einem Editor erstellt.
- Das Objektverhalten wird mit einer Programmiersprache modelliert.
- Beim Programmstart wird die Hafenumgebung geladen
- In jedem Zeitschritt
    a. werden ggf. neue Objekte erzeugt.
    b. wird die Position der Objekte neu berechnet.
    c. findet Kommunikation zwischen den Objekten statt.
    d. werden die 3D-Modelle der Objekte von der Grafikhardware an den berechneten Positionen angezeigt.

1.2 Technische Grundlagen

Die geforderten technischen Vorgänge werden von einer Game Engine optimal unterstützt. Dabei ist es ausreichend, wenn die Game Engine die grundlegenden Funktionen wie unter Anderem das Laden/Speichern von 3D-Modellen, das Ansteuern der Grafikhardware und die optimierte Durchführung der mathematischen Berechnungen ermöglicht. Spezielle Objekttypen, die üblicherweise von einer Game Engine zum Erstellen von Spielen bereitgestellt werden, sind für dieses Vorhaben oft ungeeignet, da sie in der Regel auf bestimmte (oft martialische) Spielmechaniken ausgelegt sind. Der Fokus bei der Bewertung einer Game Engine soll demnach auf den technischen Möglichkeiten liegen und weniger auf den mitgelieferten Spielobjekten.

Es ergeben sich folgende Funktionen, die eine Game Engine für dieses Teilvorhaben leisten muss:

- Szenen-Editor mit Unterstützung für benutzerdefinierte Objekte
- 3D-Modelle inklusive Animationen laden und anzeigen
- Grundlegende Datenstrukturen und Algorithmen für 3D-Berechnungen
- Datenstrukturen für nützliche Basisobjekte wie z.B. Wasser, Licht, Himmel, Kurven, positionierbare Objekte, Effekte
- Erkennung von Tastatur- und Mauseingaben
- Zweidimensionale grafische Benutzeroberflächen zur Bild- und Textdarstellung
- Physikberechnungen

Desweiteren spielt die Qualität des Programmcodes der Engine eine wichtige Rolle. Eine schlecht optimierte Engine benötigt beim Start des Programms viele Sekunden bis zum ersten Anzeigen der Szene. Dies führt in Kombination mit häufig notwendigem manuellem Testen des unfertigen Programms zu einer geringen Entwicklungsgeschwindigkeit. Die verwendete Programmiersprache hat ähnliche Auswirkungen. Moderne Hochsprachen wie C# und Java haben eine höhere Übersetzungsgeschwindigkeit zu Maschinencode als das ältere C++. Eine in C++

geschriebene Engine impliziert eine in C++ geschriebene Anwendung, die beim Übersetzen durch den Compiler oft mehrere Sekunden, im Extremfall Minuten bis Stunden braucht, während C#-Code während der Programmierung unmerklich im Hintergrund Übersetzt wird und das Starten der entwickelten Anwendung meistens sofort geschieht.

Nachfolgend werden die untersuchten 3D-Engines beschrieben.

Delta3D

- Häufig in militärischer Simulation und Trainingsanwendungen genutzt.
- Technisch nicht auf dem neuesten Stand.
- Open Source: Offenlegung des Programmcodes und somit gute Erweiterbarkeit.
- Kostenlos.
- C++: Geringere Entwicklungsgeschwindigkeit als bei C#-Engines.
- Basiert auf *OpenSceneGraph*, d.h. es können viele Dateiformate gelesen werden.
- Der logische Ablauf wird durch Ereignisse und Nachrichtenbasierte Kommunikation zwischen den Objekten gesteuert. Dadurch ist ein Protokollieren und erneutes Abspielen einer Situation möglich. Auch Netzwerkkommunikation wird dadurch vereinfacht. Die Umsetzung in C++ ist für den Entwickler aber sehr umständlich.
- Der Szenen-Editor hat nur grundlegende Funktionen und benutzerdefinierte Objekte sind nur umständlich integrierbar.
- Die Dokumentation ist mangelhaft und die Nutzergemeinde ist im Internet nicht sehr aktiv.
- Die Engine ist auf Windows und Linux lauffähig.

NeoAxis

- Wird von unabhängigen Spieleentwicklern genutzt.
- Technisch teilweise auf dem neuesten Stand.
- Kostenlos. Offenlegung des Quellcodes gegen Bezahlung.
- C#: Höhere Entwicklungsgeschwindigkeit als bei C++-Engines.
- Basiert auf der OGRE Grafikengine, welche in dem effizienteren C++ geschrieben wurde und bereits in kommerziell erfolgreichen Computerspielen zum Einsatz kam.
- Export-Plugins für die wichtigsten 3D-Modellierungsprogramme sind vorhanden.
- Der Szenen-Editor hat grundlegende Funktionen und bietet einige vordefinierte Basisobjekte, sowie eine einfache Schnittstelle für benutzerdefinierte Objekte.
- Ein Ressourcen-Editor ist vorhanden. Dieser ist für die Anpassung von Objekteigenschaften und die Erstellung von grafischen Benutzeroberflächen sinnvoll.
- Die Dokumentation ist nicht ausreichend aber die Nutzergemeinde und die Entwickler der Engine bieten im Internet Diskussionen, Anleitungen und Beispiele an.
- Die Engine ist auf Windows und im Browser lauffähig.

Unity

- Wird für kommerzielle Computerspiele im mittleren Preissegment genutzt.
- Technisch größtenteils auf dem neuesten Stand.
- Kostenpflichtig.
- Authoring-basierte Entwicklungsumgebung: Es existiert ein Editor-Programm, mit dem die Szene und das Verhalten modelliert werden.
- Gute Anbindung an 3D-Modellierungssoftware.
- Für die Programmierung wird eine Skriptsprache verwendet. Die Programme kann man zurzeit nicht schrittweise ausführen, so dass das Auffinden und Beseitigen von Fehlern (Debugging) stark beeinträchtigt ist. Dies vermindert die Entwicklungsgeschwindigkeit enorm. Zukünftige Versionen sollen das Debugging vereinfachen.
- Auf Windows, Mac OS und im Browser lauffähig.

Torque 3D

- Wird für kommerzielle Produkte im mittleren Preissegment genutzt.
- Technisch größtenteils auf dem neuesten Stand.
- Kostenpflichtig.
- Quellcode in der Professional Lizenz verfügbar.
- C++ und Skriptsprache.
- Gute Integration von 3D-Modellierungssoftware.
- Auf Windows, Mac OS und im Browser lauffähig.
- Seit Dezember 2009 verfügbar.

Unreal Development Kit (Unreal Engine 3)

- Die Unreal Engine 3 wird für hochpreisige Computerspiele verwendet. Das Unreal Development Kit ist eine vereinfachte Variante dieser Engine.
- Technisch auf dem neuesten Stand.
- Ähnlich wie die Unity Engine findet die gesamte Entwicklung in einem Editor statt. Dabei wird eine Skriptsprache genutzt, die eine Anbindung von externen Systemen wie z.B. Datenbanken nicht zulässt.
- Nur auf Windows lauffähig.
- Seit November 2009 verfügbar.

Weiter existieren auch hochpreisige professionelle Game Engines, wie z.B. die CryEngine. Diese sind oft auf konkrete Spielmechaniken ausgelegt, weil sie ein Nebenprodukt der Spieleentwicklung sind. Sie müssten für eine Simulations- und Visualisierungssoftware eventuell verändert oder erweitert werden. Die Unreal Engine (Unreal Development Kit) und die Torque 3D Engine entstammen ebenfalls einer Spieleentwicklung, sind aber bereits für den Einsatz in andersartigen Projekten diversifiziert worden.

Wegen der bereits genannten Nachteile in der Entwicklungsgeschwindigkeit wurde die Unity Engine als untauglich identifiziert. Torque 3D und das Unreal Development Kit waren zur Zeit der Entscheidung für eine Engine noch nicht verfügbar und die vermeintlich geringen Vorteile

rechtfertigen nicht den Aufwand und das Risiko einer Umstellung. Zunächst wurde für die Simulations- und Visualisierungssoftware Delta3D verwendet. Die Entwicklung ging mit NeoAxis aber etwas schneller, weshalb auf diese Engine gewechselt wurde.

1.3 Visualisierung

Die Visualisierung des simulierten Fährhafen erfolgt in Echtzeit, d.h. der Betrachtungswinkel kann jederzeit frei gewählt werden und eventuelle Auswirkungen von interaktiven Veränderungen der Simulation werden sofort sichtbar. In den folgenden Abschnitten werden die für das Teilvorhaben bedeutsamen Teile der Visualisierung beschrieben.

1.3.1 3D-Modelle

Der Rostocker Fährhafen wurde als 3D-Modell in Autodesk Maya erstellt. Als Quelle für die Modellierung dienten Pläne, Satellitenbilder, Fotos und Videoaufnahmen. Zum Übertragen des Modells von Maya in die Visualisierungssoftware wurden Programme der Engine und selbstentwickelte Konverter verwendet.

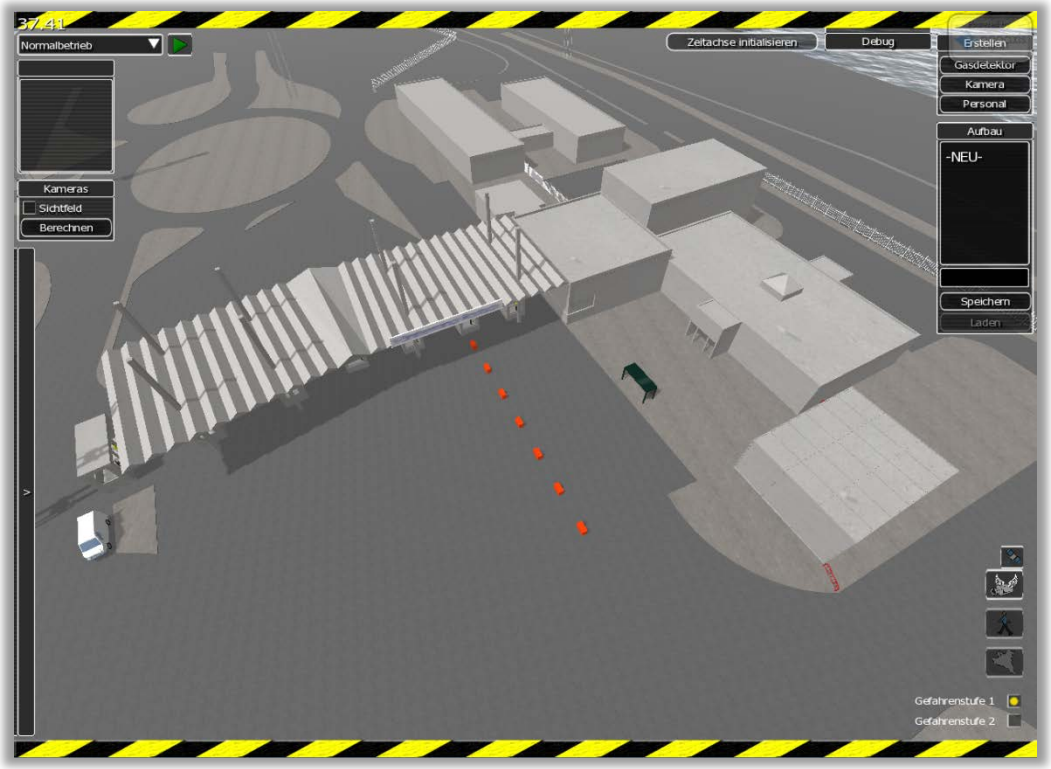

Abb. 2: Nachbildung der Gebäude des Rostocker Fährhafens

Die Fähre Mecklenburg-Vorpommern wurde als 3D-Modell in Autodesk Maya erstellt. Als Quelle für die Modellierung dienten Pläne, Fotos und Videoaufnahmen. Das Modell wird im virtuellen Fährhafen und im EUS-Demonstrator genutzt. Für den EUS-Demonstrator wurden spezielle Anforderungen an das Modell gestellt. Zum Beispiel müssen alle Decks voneinander trennbar sein.

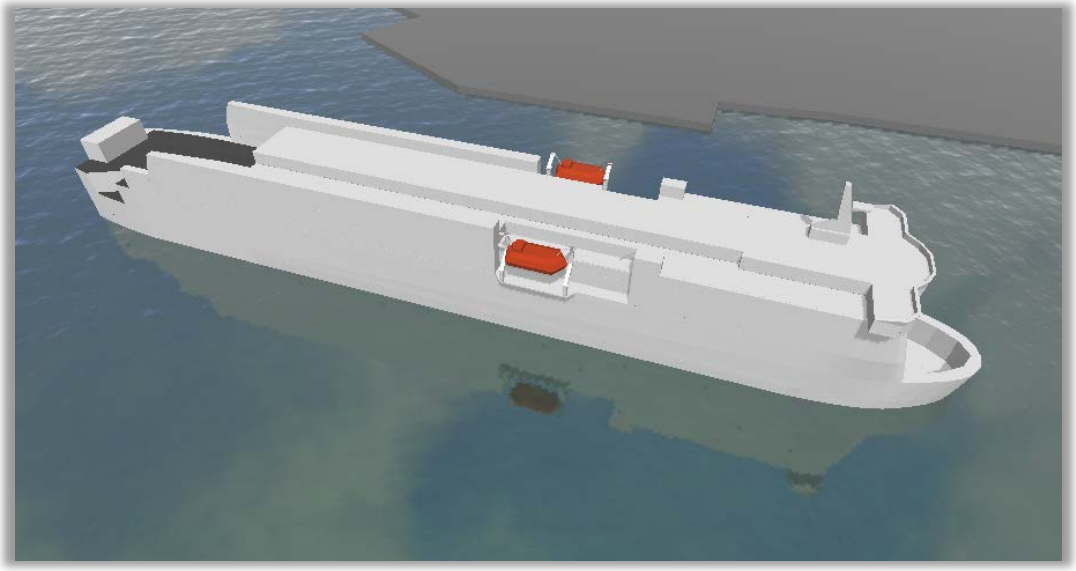

Abb. 3: Nachbildung der Fähre Mecklenburg-Vorpommern

Für die Personen wird das Paket *Complete Characters* von Rocketbox[1] verwendet. Es enthält 104 3D-Modelle von Personen unterschiedlichen Geschlechts und Alters mit ziviler Bekleidung in drei Detailstufen. Die Modelle wurden mit Autodesk 3D Studio Max und selbstentwickelten Programmen in die entstandene Software überführt.

Abb. 4: Beispiele aus "Complete Characters" von Rocketbox[1]

Für die Fahrzeuge werden hochauflösende 3D-Modelle von DOSCH DESIGN[2] verwendet. Während der Entwicklung wurden selbsterstellte detailarme Platzhalter verwendet, um die Ausführungsgeschwindigkeit zu erhöhen.

[1] www.rocketbox-libraries.com
[2] www.doschdesign.com

Abb. 5: Beispiel aus „Cars 2005 V1.1“ von DOSCH DESIGN[2]

1.3.2 Sicherheitsrelevante Gegenstände

Die Passagiere können sicherheitsrelevante Gegenstände mit sich führen, welche bei einer Kontrolle durch das Sicherheitspersonal oder das Gasdetektorsystem erkannt werden. Zur Visualisierung der (in der Realität meist verborgenen) Gegenstände werden Symbole verwendet.

Abb. 6: Sprengstoffsymbol über einem Attentäter

Beim Angriff kommt es je nach verwendetem Gegenstand zu einem grafischen Effekt und zur Darstellung der davon beeinflussten Passagiere.

1.3.3 Gasdetektorsystem

Zur Visualisierung der Funktionsweise eines Gasdetektorsystems wurde ein virtuelles Modell nach den Vorgaben des Fraunhofer SDF erstellt. Hierbei handelt es sich um einen tunnelförmigen Gang, in dem ein Luftstrom quer zur Laufrichtung der Passagiere verläuft und eventuelle Ausgasungen von gefährlichen Stoffen zu einem Halbleitersensor transportiert. Dieser Sensor meldet die Detektionsergebnisse an einen Mitarbeiter, der die Passagiere anschließend manuell durchsuchen kann.

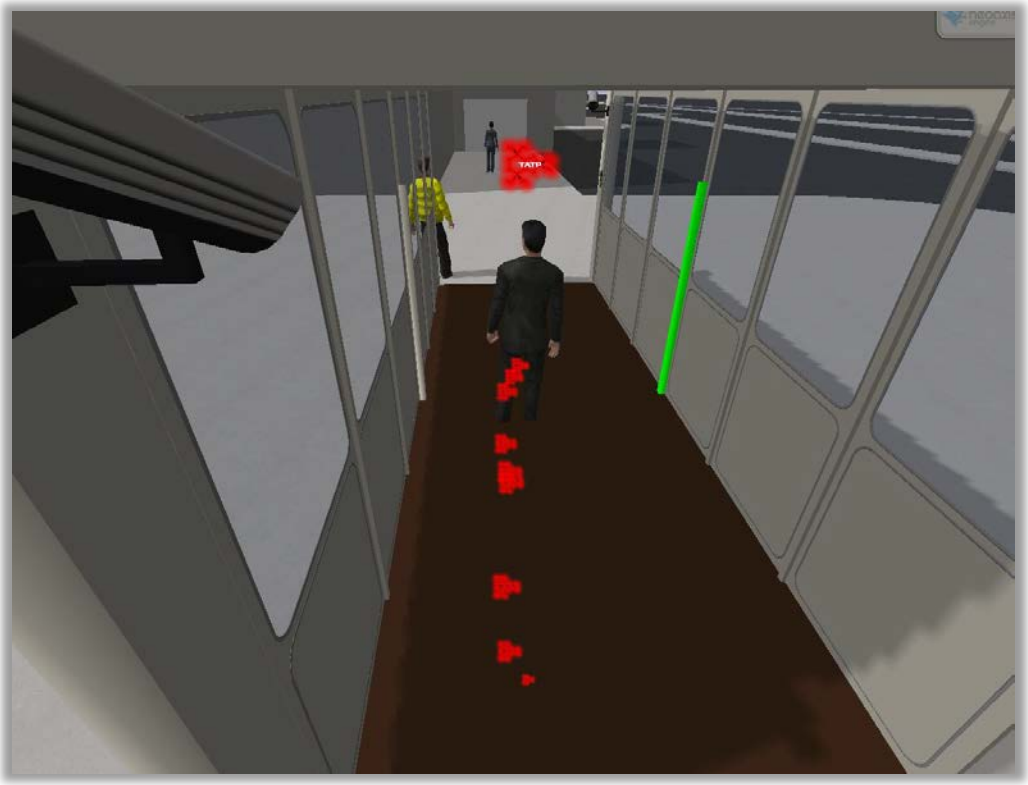

Abb. 7: Blick durch den Tunnel des Gasdetektorsystems während ein Passagier mit Sprengstoff erfasst wird. Die Moleküle werden als farbige chemische Strukturformeln dargestellt. Der Sensorzustand wird durch Ampelfarben (hier: grün) visualisiert.

Zum besseren Verständnis der Funktionsweise werden die Ausgasungen der mitgeführten Stoffe als übernatürlich große Moleküle dargestellt. Der Proband zieht eine Molekülspur hinter sich her, welche in der Mitte des Tunnels seitlich abgelenkt wird, so dass sie auf den Sensor trifft. Die entstehenden Sensorwerte werden als Farbverlauf dargestellt. Für die Molekülbewegung und den zeitlichen Verlauf der Sensorwerte kamen die Simulationsergebnisse des Fraunhofer SDF zum Einsatz.

1.3.4 Überwachungskameras

Im virtuellen Fährhafen können beliebig viele Kameras an beliebigen Orten aufgestellt werden. Der Blickwinkel jeder Kamera wird durch die Projektion der jeweiligen Sichtebene auf die, sich in Blickrichtung befindliche, Geometrie dargestellt. Zur genaueren Analyse kann der Anteil der begehbaren Flächen, welcher von mindestens einer Kamera überwacht wird, ermittelt werden. Damit wird die Komposition der Überwachungskameras auf dem nachgebildeten echten Hafengelände objektiv bewertbar und lässt sich durch wenige, einfache Handgriffe verbessern.

Abb. 8: Überwachungskamera (rechts oben, rot markiert) inkl. Sichtbereich (grün) und Kamerabild im Fenster rechts unten.

1.4 Simulation

Ziel der Simulation ist es, reale Vorgänge möglichst detailgetreu in der virtuellen Welt abzubilden, um ein System in unterschiedlichen Situationen zu analysieren. Bei besonders komplexen Systemen, wie dem Sicherheitssystem eines Hafens, kann die theoretische Betrachtung der Effizienz nicht mehr für alle Situationen durchgeführt werden.

Ein Ansatz zur Simulation ist die Vorgabe eines Szenarios aufgrund dessen der Durchsatz verschiedener Akteure im System modelliert wird. Dies ähnelt der Modellierung eines Geschäftsprozesses. Hierfür existieren bereits Modellierungs- und Simulationsverfahren in Form von professioneller Software (z.B. iGrafx). Eine Visualisierung der Ergebnisse ist jedoch nicht trivial. Betrachtet man lediglich die Bewegung von Personen von Punkt A nach Punkt B auf dem Hafengelände, so besagt die Simulation, dass diese Bewegung in der Zeit t absolviert wird. Darin sind bereits alle modellierten Faktoren, etwa das Ausweichen anderer Personen und das Warten an roten Ampeln, enthalten. Bei einer Visualisierung sollen diese Details aber auch angezeigt werden. Der exakte Laufweg von Personen muss also aus den aggregierten Prozessgrößen berechnet werden. Im Beispiel bietet die Zeit t dafür nicht ausreichend Informationen. Für die Analyse eines Sicherheitssystems ist die Methode also ungeeignet. Experimente mit der Integration von Simulationen aus iGrafx bestätigten diesen Eindruck.

Eine alternative Methode wäre das Aufstellen eines sehr detaillierten Modells und die Bewahrung aller Zwischenergebnisse der Simulation. Dies lässt sich mit 3D-Animationssoftware erstellen und zudem noch visualisieren. Eine Veränderung der Ausgangsbedingungen zieht dann aber eine Anpassung aller davon abhängigen Objekte mit sich, so dass die Parametrisierung der Szenarien nicht möglich ist.

Der umgekehrte Ansatz wäre die feingranulare (physikbasierte) Simulation einzelner Objekte. Jedes Objekt hätte ein eigenes Verhalten, das ggf. von anderen Objekten abhängig ist. Die Kombination der einzelnen Aktionen bildet das Systemverhalten (Szenario). Dieser Ansatz erfordert eine geschickte Feinabstimmung der Parameter, damit letztendlich das gewünschte Szenario entsteht. Für die Simulation menschlicher Intelligenz gibt es abstrakte Modelle, welche die Wirklichkeit abbilden sollen, so dass realistische Szenarien simuliert werden können. Der feingranulare Ansatz ist stark objektorientiert und lässt sich durch eine objektorientierte Programmiersprache (C++, C#, Java, Python, …) optimal umsetzen. Die Entscheidung fiel daher auf diesen Ansatz.

Zur Realisierung bieten sich drei Technologien an:

- Die Programmierung aller Simulationsroutinen mit einer Programmiersprache und das damit verbundene statische Einbinden in das Gesamtsystem. Endanwender können die Simulation hierbei nur eingeschränkt verändern.
- Die Programmierung aller Simulationsroutinen mit einer Skriptsprache und das damit verbundene dynamische Laden der Objektverhaltensweisen aus veränderbaren Dateien. Ein Endanwender muss die Skriptsprache beherrschen, um die Simulation zu verändern. Eine Skriptsprache ist leichter zu erlernen als eine Programmiersprache, aber das Einbinden ist relativ Aufwändig und Risikoreich und muss daher mit den potentiellen Vorteilen abgewogen werden.
- Das Modellieren aller Simulationsroutinen durch Aneinanderreihung von Logikbausteinen in einem speziell dafür angefertigten Softwarewerkzeug. Hierfür werden vom Endanwender keine speziellen Kenntnisse erfordert, aber der Funktionsumfang wird im Vergleich zu Programmier- und Skriptsprachen stark eingeschränkt und die Maßanfertigung eines solchen Werkzeuges ist sehr aufwändig.

Im Hinblick auf zukünftige Erweiterungen wurde das Objektverhalten stark abstrahiert, sodass alle simulationsrelevanten Objekte aus einem einfachen Zustandsautomat bestehen. Die Verwendung einer Skriptsprache ist daher kaum von Vorteil. Der Einsatz von selbstentwickelten Modellierungswerkzeugen erschien unverhältnismäßig, da eventuelle Weiterentwicklungen aufgrund des Forschungscharakters nicht abgeschätzt werden können. Nach Projektende wird der Quellcode an die zuständigen Projektpartner weitergegeben, sodass nachträgliche Änderungen vorgenommen werden können.

Die entstandene Anwendung dient neben der Simulation von Zugangskontrollen auch der Visualisierung von neuartigen Kontrolltechniken. Dabei kann es vorkommen, dass die Simulation verlangsamt oder wiederholt werden muss, um einen Vorgang zu präsentieren bzw. zu verstehen. Eine Game Engine läuft üblicherweise in Echtzeit, weshalb für den virtuellen Fährhafen ein selbstentwickeltes Zeitmanagement angewendet wird. Damit lässt sich die Simulation jederzeit verlangsamen oder beschleunigen. Das Wiederholen eines Simulationsabschnitts wird durch das Persistieren der Objektzustände in kurzen Zeitabschnitten erreicht. Ein Endanwender kann dadurch zu jedem beliebigen Zeitpunkt in einer bereits durchgeführten Simulation zurückspringen.

1.4.1 Objektverhalten

Im Projektverbund wurde eine Prozesserfassung auf einem echten Hafengelände durchgeführt und daraus ein Modell gebildet, das die Vorgänge bei der Abfertigung von Personen und Fahrzeugen im Normalbetrieb beschreibt. Dieses Modell wurde für die Entwicklung der Simulation herangezogen. Alle simulationsrelevanten Objekte werden als Zustandsautomat simuliert. Unter dieser sehr abstrakten Verhaltensbeschreibung liegen weitaus komplexere Algorithmen zur Simulierung von realitätsnahem Verhalten:

Personen

Für die künstliche Intelligenz von Menschen wurde ein Schichtenmodell verwendet, bei dem das Verhalten von mehreren eigenständigen Berechnungseinheiten simuliert wird. Die einzelnen Schichten sind nach ihrer Komplexität geordnet. An oberster Stelle steht ein Algorithmus zum Erzeugen rationaler Entscheidungen. In dieser Einheit werden der eigene Zustand und die Umgebung ausgewertet, um ein grobes Ziel zu definieren. Diese Entscheidung wird dann von der nächsten Schicht in eine grobe Wegbeschreibung umgewandelt. Die darauffolgende Schicht erzeugt aus der groben Wegbeschreibung den kürzesten Weg. So wird die Information der rationalen Entscheidung schrittweise verfeinert, bis letztendlich die aktuelle Position der Person feststeht.

Das verwendete Schichtenmodell besteht aus zehn Schichten:

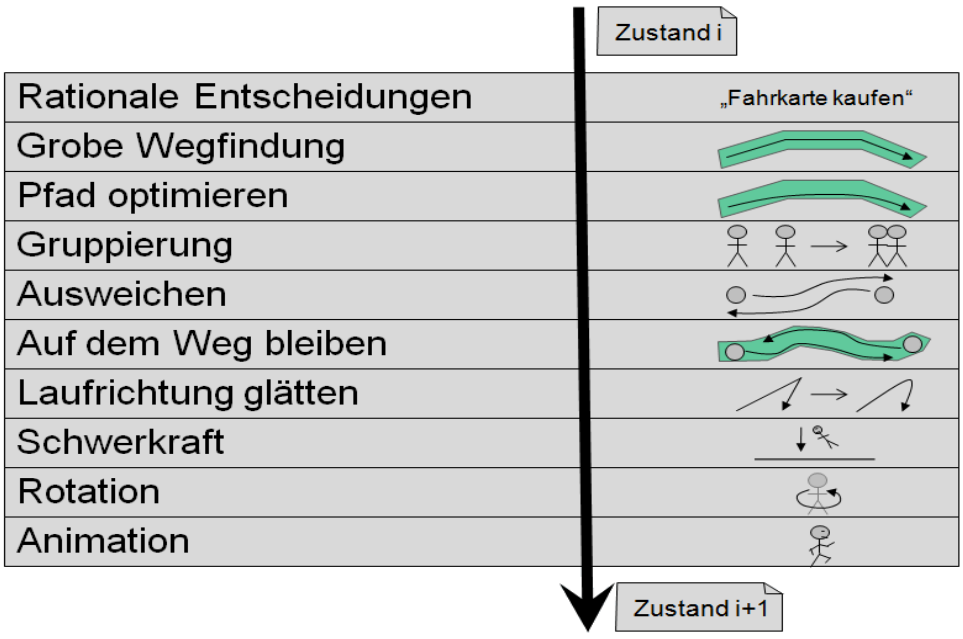

Abb. 9: Schichtenmodell der Personen-KI

Rationale Entscheidungen

Ein Passagier kann unterschiedliche Ziele verfolgen: in ein Fahrzeug einsteigen, zu einem Ziel gehen, einen gefährlichen Gegenstand anwenden, für eine Durchsuchung stehen bleiben, etc. Jedes Bewegungsziel ist mit einem Ort im Hafen verbunden. Alle Orte sind untereinander durch Fußwege und Personenbeförderungsmittel vernetzt. Bei einer Änderung des aktuellen Ziels wird ein Pfad zum Bestimmungsort generiert. Beispiel: „Reedereibüro – Bushaltestelle – Bus – Bushaltestelle – Wartebereich – Fähre“. Es entsteht eine Liste von Orten, aber noch keine genaue

Bewegungsrichtung. Die Verbindungswege zwischen den Orten werden in der nächsten Schicht gesucht.

Grobe Wegfindung

Alle begehbaren Flächen im Hafen sind durch drei- bzw. viereckige Polygone markiert (*navigation mesh).* Dafür wurde im Szenen-Editor ein Netz aus Punkten angelegt, zwischen denen die Polygone aufgespannt werden. Aus der übergeordneten Schicht ist der nächste Bestimmungsort bekannt (Im Beispiel: Reedereibüro). Mit diesem Ort und der aktuellen Position wird über den A*-Algorithmus[3] eine Polygonliste generiert, die den groben Weg beschreibt. Hier wird also festgelegt, welche Fußwege die Person verwendet.

Pfad optimieren

Die Polygone sind mitunter mehrere Meter breit, so dass der genaue Weg noch nicht feststeht. In dieser Schicht wird deshalb der kürzeste Weg auf den Polygonen mit dem Funnel-Algorithmus[4,5] berechnet. Der Weg ist zwar optimal, enthält aber spontane Richtungswechsel und verläuft oft auf den Kanten des Fußweges. Das wirkt unnatürlich und wird deshalb von den nachfolgenden Schichten verändert.

Gruppierung

Bevor der Pfad weiter optimiert werden kann, müssen Interaktionen mit anderen Fußgängern beachtet werden. Ein natürliches Phänomen ist die Gruppierung von Personen. Nach einer Untersuchung des Trinity Colleges in Dublin[6] treten Gruppierungen von zwei Personen mit 46% am häufigsten auf, gefolgt von einzelnen Personen mit 37% und Dreiergruppen mit 11%. Größere Gruppen treten selten auf. Die Untersuchung fand auf dem Gelände der Universität statt. Für Fährhäfen liegen keine Untersuchungen vor, aber aufgrund der geringen Anzahl an Fußpassagieren (ca. 20 pro Fahrt) erscheinen die Zahlen zweckmäßig. Die Personen bewegen sich innerhalb der Gruppe in einer Formation, die von der Gruppierungsschicht eingehalten wird.

Ausweichen

In der Realität weichen die Menschen einander aus, wenn sie sich entgegen kommen oder mit unterschiedlicher Geschwindigkeit gehen. Um dieses Verhalten zu simulieren wird in jedem Zeitschritt der Raum vor der Person auf andere Fußgänger untersucht. Steht eine Kollision bevor, so wird ein Ausweichmanöver nach links oder rechts eingeleitet, je nachdem, welche Richtung den kürzeren Umweg bedeutet.

Auf dem Weg bleiben

Durch die Interaktion mit anderen Fußgängern kann die Person vom Weg abgelenkt werden. In der Realität arrangieren sich die Personen so, dass niemand den Weg verlässt. Dies wird von dieser Schicht durch eine Kraft simuliert, die die Person von der Kante in Richtung Mitte des Weges drückt. Das verhindert auch die unnatürliche Bewegung auf der Kante eines Weges, die von der Pfadoptimierungsschicht erzeugt wurde.

---

[3] Hart, P.E.; Nilsson, N.J.; Raphael, B. (1968). *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*. IEEE Transactions on Systems Science and Cybernetics, vol.4, no.2, pp.100-107

[4] Demyen, D. J. (2006). *Efficient Triangulation-Based Pathfinding.* University of Alberta, pp.52-57, http://www.cs.ualberta.ca/~mburo/ps/thesis_demyen_2006.pdf

[5] Miles, D. (2006). *Crowds In A Polygon Soup Next-Gen Path Planning.* NavPower, GDC 06, http://www.navpower.com/gdc2006_miles_david_pathplanning.ppt

[6] Peters, C.; Ennis, C. (2009). *Modeling Groups of Plausible Virtual Pedestrians*. IEEE Computer Graphics and Applications, vol.29, no.4, pp.54-63

Laufrichtung glätten

Durch die Optimierung des Pfades versuchen die Personen Kurven möglichst eng zu schneiden. Dies resultiert in einer spontanen Richtungsänderung, die in der Realität nicht vorkommt. Durch eine Glättung der Bewegungsrichtung zu jedem Zeitschritt wird dies verhindert.

Schwerkraft

Alle übergeordneten Schichten stellen Ihre Berechnungen im zweidimensionalen Raum an. Die dritte Dimension ist bei Fußgängern nur von der Schwerkraft bestimmt. Sie drückt die Personen auf den Boden. Dabei ist darauf zu achten, dass Höhenunterschiede nur bis zu einer gewissen Höhe überwunden werden können.

Rotation

Die Person wird in Laufrichtung gedreht. In der Simulation läuft eine Person niemals seitwärts oder rückwärts.

Animation

Auf Basis der Geschwindigkeit wird eine Animation ausgewählt. Stehende Personen werden durch eine wartende Haltung dargestellt. Sich bewegende Personen werden je nach Geschwindigkeit gehend oder rennend dargestellt. Personen in Fahrzeugen sitzen und kommunizierende Personen bewegen Kopf und Arme. Die nötigen Informationen stammen ausschließlich aus den übergeordneten Schichten.

Fahrzeuge

Das Verhalten der Fahrzeuge ist zunächst sehr einfach zu beschreiben: Es existieren virtuelle Linien in der Mitte der Straße, auf denen sich die Fahrzeuge wie auf Schienen bewegen können. An Kreuzungspunkten sind alle Linien der betreffenden Straßen verbunden. Alle Fahrspuren sind nur in einer Richtung befahrbar. Dies führt zu einem einfachen Modell für die Kreuzungen. Fahrspuren können sich entweder aufteilen oder zusammenlaufen. Komplizierte Vorfahrtsregeln können dadurch ausgeschlossen werden.

Bei der Aufteilung einer Fahrspur brauchen ankommende Fahrzeuge nicht darauf achten, ob andere Fahrzeuge in der Nähe der Kreuzung sind. Bei zusammenführenden Kreuzungen ist die Situation komplexer. Es kann vorkommen, dass mehrere Fahrzeuge aus unterschiedlichen Richtungen auf eine Fahrspur treffen. Dabei würde es ohne weitere Vorsichtsmaßnahmen zu Unfällen kommen. Die ankommenden Fahrzeuge müssen also auf die anderen Fahrspuren achten. Dafür kann man von der Netzstruktur des Straßennetzes profitieren.

Jedes Fahrzeug verfolgt seine eigene Fahrspur bis zur Kreuzung und sucht dann auf allen anderen eintreffenden Fahrspuren einige Meter in umgekehrte Richtung nach anderen Fahrzeugen. Wenn ein anderes Fahrzeug gefunden wurde, wird auf eine mögliche Kollision innerhalb des Bremsweges beider Fahrzeuge geprüft. Das Fahrzeug, welches auf das andere Fahrzeug auffahren würde, wird verlangsamt.

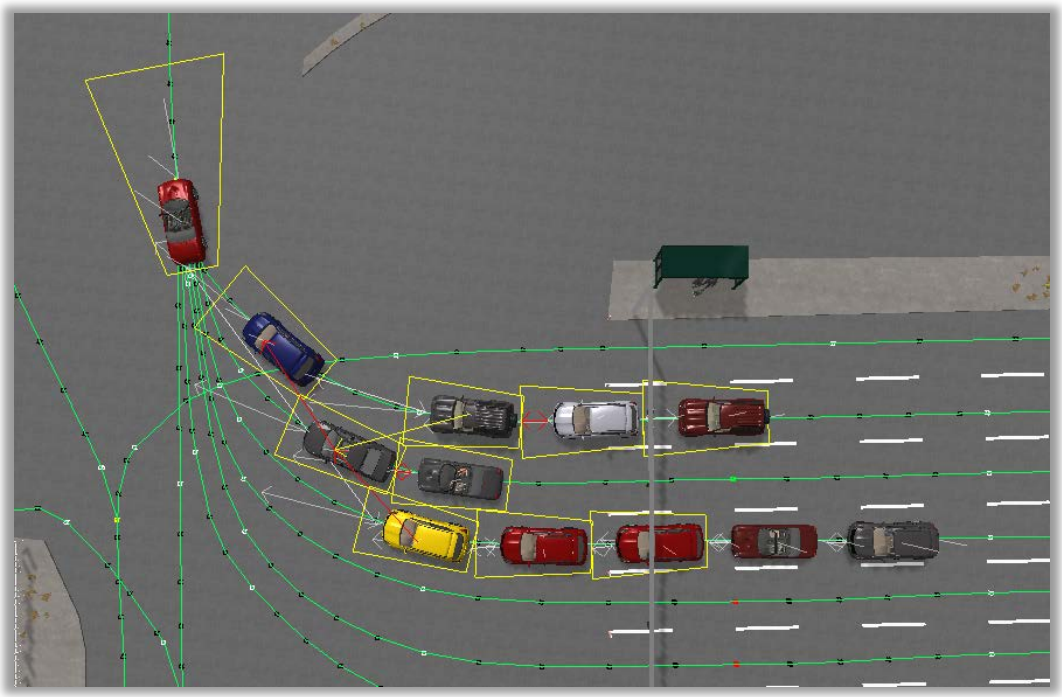

Abb. 10: Fahrzeuge beim Zusammenführen mehrerer Fahrbahnen. Die Kollisionsprüfung findet in den gelben Bereichen statt. Rote Pfeile markieren das Warteverhalten.

Auf einfachen Fahrspuren wird lediglich auf das vorausfahrende Fahrzeug geachtet. Unterschreitet der Abstand den Bremsweg des hinteren Fahrzeuges, dann wird es abgebremst. Die Erkennung von vorausfahrenden Fahrzeugen muss nicht in jedem Zeitschritt durchgeführt werden. Es reicht aus, wenn bei jeder Überquerung einer Kreuzung das vorausfahrende und das verfolgende Fahrzeug gesucht und gespeichert werden. So entsteht eine Art Bekanntheitsgraph, der für fahrzeugübergreifende Algorithmen genutzt werden kann.

1.4.2 Szenarios

Im Projektverbund wurden bestimmte Szenarios definiert, die im virtuellen Fährhafen umgesetzt wurden. Solche Szenarios basieren in der Regel aus einfachen Ereignissen, wie etwa das Erscheinen eines Passagiers (zu Fuß oder im Fahrzeug) und die anschließende Verbringung eines gefährlichen Gegenstands auf das Schiff oder in die Hafenanlagen. Demnach können weitere Szenarios durch die Parametrisierung dieser grundlegenden Vorgänge erstellt werden. Hierfür wurde ein Szenarioeditor angefertigt, mit dem der zeitliche und räumliche Verlauf der Aktionen definiert werden kann.

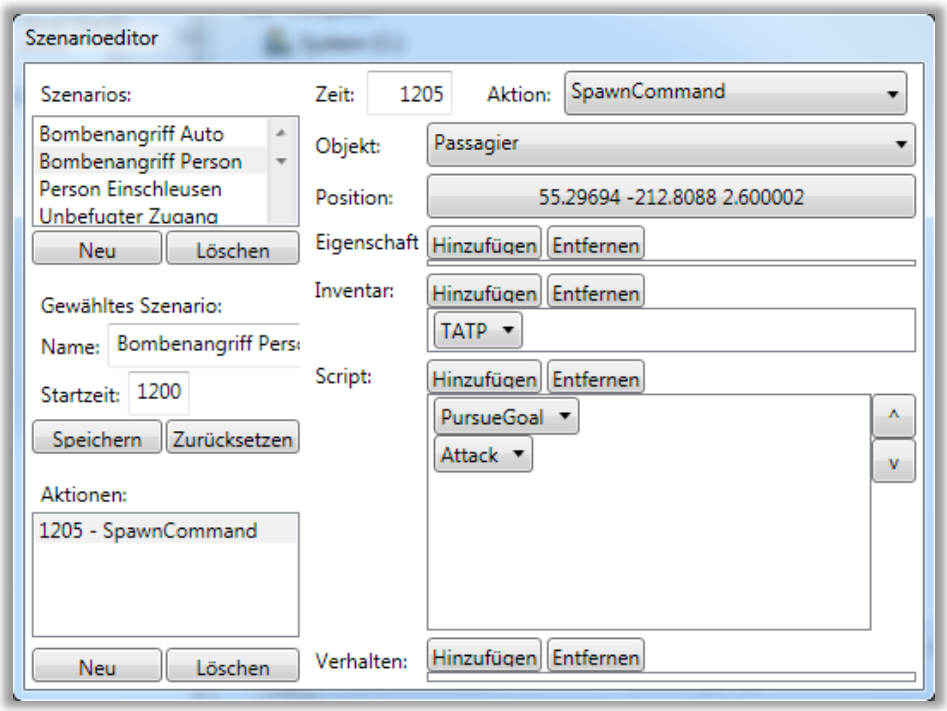

Abb. 11: Szenarioeditor als separates Fenster

1.4.3 Ergebnisse

Während der Simulation werden bestimmte Ereignisse aufgezeichnet und nach Simulationsende in einer XML-Datei gespeichert. Zu den aufgezeichneten Ereignissen gehören hauptsächlich das Erzeugen und Löschen von Objekten und die Dauer ihrer Bewegung zwischen den einzelnen Stationen auf dem Hafengelände. Eine einfache Visualisierung bietet eine grobe Übersicht über den Passagierdurchsatz und kann zum Vergleich verschiedener Sicherheitsarchitekturen verwendet werden.

2 EUS-Demonstrator

Für die Arbeitspakete 4.1.2, 4.1.3 und 4.2 entstand ein Demonstrator für ein Entscheidungsunterstützungssystem im Bereich der Security auf Fährschiffen. In Zusammenarbeit mit dem Fraunhofer FKIE wurden neuartige Wege der Informationsvisualisierung und Mensch-Maschine-Interaktion identifiziert und anschließend vom GITZ im Rahmen des Demonstrators umgesetzt. Es entstand ein Anforderungskatalog an die Informationsvisualisierung und eine Einschätzung der Integrationsfähigkeit der entwickelten Komponenten in bestehende Systeme.

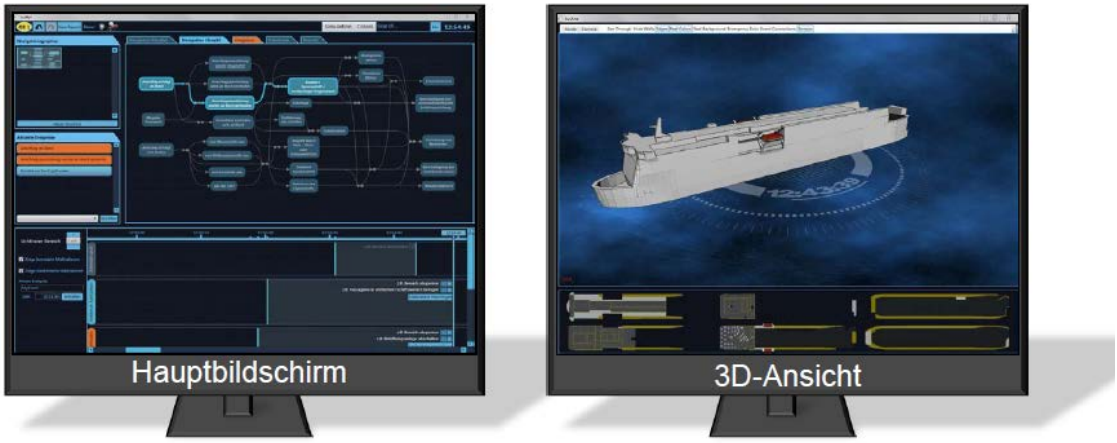

Abb. 12: Die zwei Bildschirme des EUS-Demonstrators.

2.1 Anforderungen

Siehe Anlage A2.

2.2 Technische Grundlagen

Zur Realisierung zweidimensionaler Benutzeroberflächen eignen sich *window toolkits* oder *Grafik-Frameworks*. Diese Softwarebibliotheken bieten eine Vielzahl an vorgefertigten grafischen Elementen an, die zum Teil aus Kompositionen zu noch komplexeren Elementen bestehen. Das bekannteste Werkzeug zur Erstellung grafisch anspruchsvoller Bedienoberflächen ist Adobe Flash. Seit 2006 ist ein ähnliches Produkt von Microsoft verfügbar: Windows Presentation Foundation (WPF) und dessen Browservariante Silverlight.

Adobe Flash erlangte aufgrund der Plattformunabhängigkeit als Browserplugin große Beliebtheit. Mit der eigenen Programmiersprache *Action Script* und das internetorientierte Programmiermodell lassen sich Datenbanken und andere entfernte Server anschließen. Eine Verbindung mit Schiffssystemen, wie etwa Sensoren oder Kommunikationsanlagen, ist hingegen schwierig. Die Darstellung dreidimensionaler Objekte wurde zu Beginn des Teilvorhabens nicht durch die Grafikhardware beschleunigt, sodass Adobe Flash für die Entwicklung des EUS-Demonstrators ungeeignet war.

Microsoft Windows Presentation Foundation basiert auf dem .Net Framework, welches gute Möglichkeiten zum Anschluss an Systeme aller Art bietet. So kann auch die Verbindung mit einer dreidimensionalen Darstellungskomponente sehr einfach realisiert werden. Wenn diese Darstellungskomponente ebenfalls auf dem .Net Framework basiert, ist die Kommunikation zwischen den beiden Komponenten trivial. Zum Darstellen einer 3D-Welt mit dem .Net Framework bieten sich mehrere Grafik-Engines an. Im EUS-Demonstrator wird Microsoft XNA verwendet, weil dies sehr nahe an der Funktionalität der Grafikhardware arbeitet und dem Entwickler somit viele Freiheiten lässt. Zusatzfunktionen wie Audioausgabe, Tastatur- und Mauseingabe und Algorithmen für 3D-Berechnungen sind im ausreichenden Maß vorhanden.

2.3 Arbeitsdomäne

In Abstimmung mit dem Fraunhofer FKIE fiel die Wahl auf eine Lösung mit zwei Bildschirmen. Auf dem einen Bildschirm finden ein Großteil der Benutzereingaben und die Darstellung der abstrakten Informationen statt (Hauptbildschirm). Der andere Bildschirm dient der Visualisierung des Schiffszustands und der räumlichen Informationen der Ereignisse (3D-Ansicht).

Die Informationsmenge auf dem Hauptbildschirm ist zu groß, um sie gleichzeitig darzustellen bzw. zu erfassen. Darum wurden gleichzeitig benötigte Informationen logisch gruppiert und Tab-Fenstern zugeordnet (vergleichbar mit Tabs in aktuellen Internet-Browsern). Es existieren folgende Tabs:

- *Navigation*: Eingabe der aufgetretenen sicherheitsrelevanten Ereignisse mittels eines Entscheidungsgraphs.
- *Ereignisse*: Liste der aktuellen Ereignisse inklusive Zusatzinformationen und einer Liste von Maßnahmen, die zur Verbesserung der Sicherheit beitragen.
- *Datenbank*: Prototypenhafte Einbindung einer Informationsdatenbank.
- *Reports*: Erstellte Berichte über Ereignisse oder Situationen.

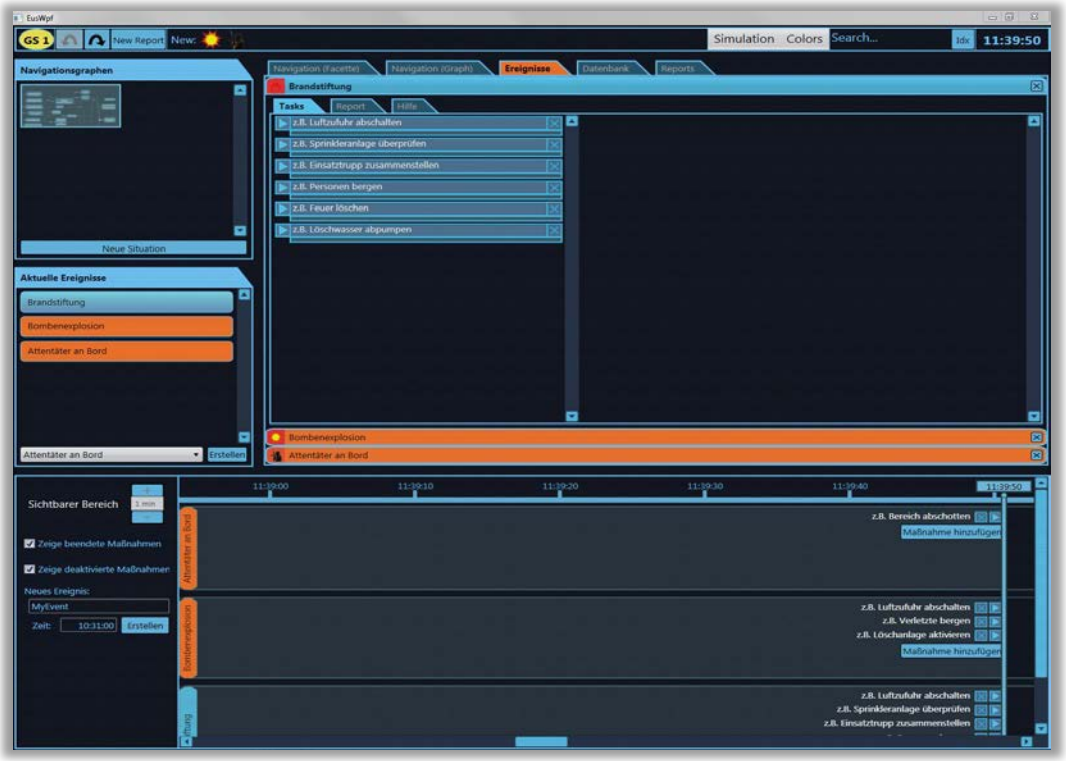

Abb. 13: Linker Bildschirm mit drei ausgewählten Ereignissen. Oben/Mitte: Tab-Bereich mit aktiver Ereignisansicht; Unten: Zeitleiste; Links: Auflistung der aktiven Ereignisse.

Der zeitliche Verlauf der Gefahrensituation wird dauerhaft im unteren Bildschirmbereich dargestellt. Hierfür wurde eine Zeitachse gewählt, an der die aufgetretenen Ereignisse und die durchgeführten Maßnahmen aufgetragen werden.

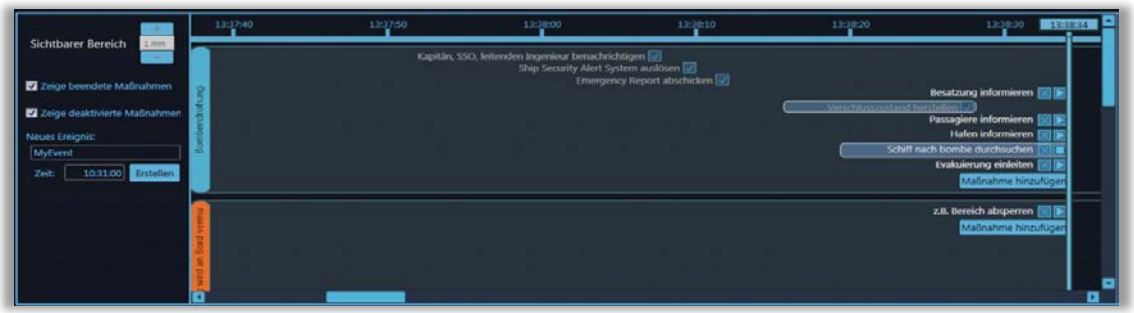

Abb. 14: Zeitlicher Verlauf der Ereignisse.

Informationsübergreifende Techniken zur Sicherstellung der effizienten Informationsübertragung in der Mensch-Maschine-Schnittstelle sind beispielsweise die unterschiedlichen Farbschemata und grafische Effekte zur Verstärkung des Sinneseindrucks bei Interaktion (Feedback).

2.4 Situationserkennung

Die Repräsentation einer Gefahrensituation im EUS-Demonstrator ist die Menge der kürzlich aufgetretenen Ereignisse. Jedes Ereignis beschreibt die Aktion eines Angreifers oder präzisiert diese. Ereignisse bestehen aus einer textuellen Beschreibung, einer Liste von Maßnahmen zur Wiederherstellung der Sicherheit und aus zusätzlichen Informationen, wie z.B. allgemeine Verhaltensregeln der Besatzung im Schadensfall.

Für die Erkennung der Gefahrensituation muss der Mensch sorgen, da Security-bedingte Ereignisse oft nicht von Sensoren erfasst werden können. Der Informationseingabe in der Mensch-Maschine-Schnittstelle wird daher große Bedeutung zugemessen. In Kooperation mit dem Fraunhofer FKIE entstanden zwei Realisierungsalternativen, die auf einem Entscheidungsgraphen basieren. Ein Entscheidungsgraph besteht im EUS-Demonstrator aus einer Menge von Ereignissen, die untereinander durch kausale oder temporale Zusammenhänge verbunden sind. Ein Pfad in diesem Graph beschreibt eine Situation oder einen in sich geschlossenen Teil der Gesamtsituation. Nachfolgend werden zwei Visualisierungstechniken beschrieben.

2.4.1 Gerichteter Graph

Die Darstellung des Entscheidungsgraphen (auch Kontextgraph genannt) kann auf herkömmliche, aus der Literatur bekannte, Weise geschehen. Hierfür werden die Knoten (Ereignisse) des Graphen als Rechteck/Ellipse dargestellt und die Kanten (kausale Zusammenhänge) als Linien zwischen den Knoten. Der genutzte Beispielgraph bestand aus ca. 20-30 Knoten. Der entscheidende Faktor für die schnelle Erkennung und Navigation des Anwenders in dem Graphen ist das Layout. Die Knoten von logisch zusammenhängenden Teilgraphen stehen idealerweise in räumlicher Nähe zueinander. Die Anzahl der kreuzenden Kanten ist zu minimieren. Für diese Aufgabe existieren bereits Algorithmen, die hier verwendet werden können.

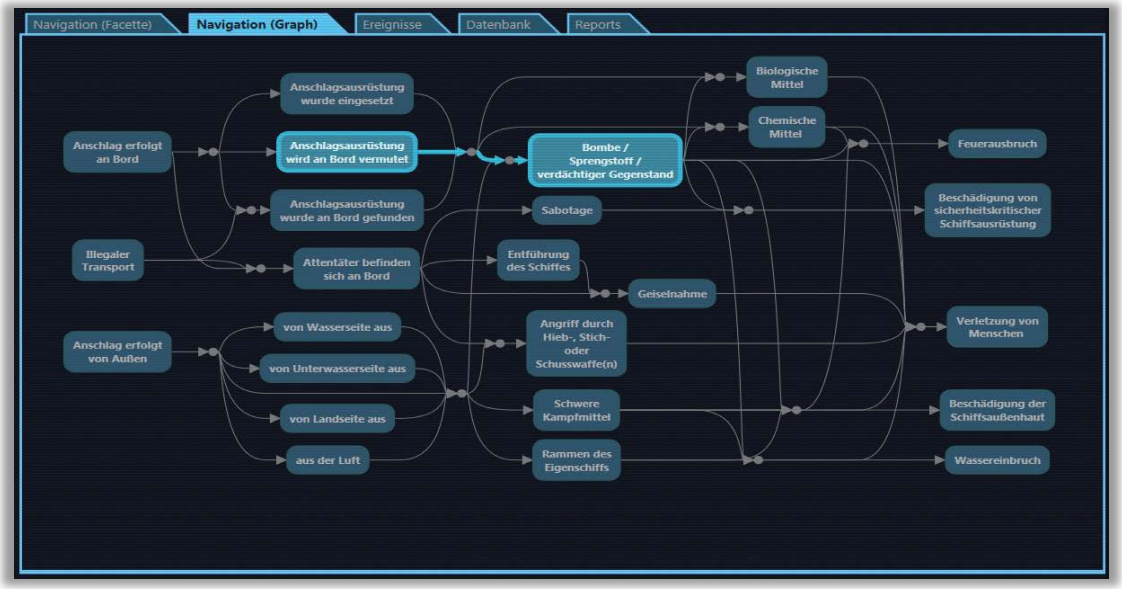

Abb. 15: Entscheidungsgraph mit ausgewählten Ereignissen.

Softwarebibliotheken wie GraphViz und OGDF bieten eine Vielzahl an Layout-Algorithmen für planare Graphen und Bäume. Da der Entscheidungsgraph jedoch nicht planar und (in der ursprünglichen Fassung) kein Baum war, müssten zunächst einige Kanten entfernt und nach dem Berechnen des Layouts wieder hinzugefügt werden. Ein spezieller Algorithmus für nicht-planare Graphen, der in der näheren Auswahl stand, ist der Fruchterman-Reingold-Algorithmus. Dieser wendet eine physikalische Simulation auf ein Feder-Masse-System an und erreicht dadurch einen Zustand mit minimaler Kantenlänge und –überdeckung. Allerdings konnte keiner der geprüften Algorithmen die Knoten inhaltlich gruppieren, so dass letztendlich ein manuelles Layout die besten Ergebnisse lieferte. Daher wurde ein Editor entwickelt, mit dem das Layout einmalig erstellt und gespeichert werden kann.

Wegen der großen Anzahl an Knoten im Graphen bei gleichzeitig sehr beschränktem Bildschirmplatz wurde vom Fraunhofer FKIE ein neues Visualisierungskonzept vorgeschlagen, welches im nächsten Abschnitt beschrieben wird.

2.4.2 Facettierte Taxonomie

Die Visualisierung der facettierten Taxonomie mit dem Faceted Search Browser gleicht der eines Programmenüs. Anfangs ist eine grobe Auswahl an Ereignissen sichtbar. Wenn Ereignisse ausgewählt werden, erscheinen Untermenüs, mit denen das jeweilige Ereignis präzisiert werden kann. Ein entscheidender Vorteil dieser Darstellung gegenüber der Visualisierung des Graphen ist das Verstecken von unwichtigen Informationen und die damit verbundene Platzeinsparung. Die zugrundeliegende Datenstruktur ist ein Baum. Die Gesamtsituation ergibt sich aus der Kombination von gewählten Knoten. Für viele Kombinationen existieren beispielhafte Ereignisse mit Maßnahmen, um die Funktionsweise zu demonstrieren.

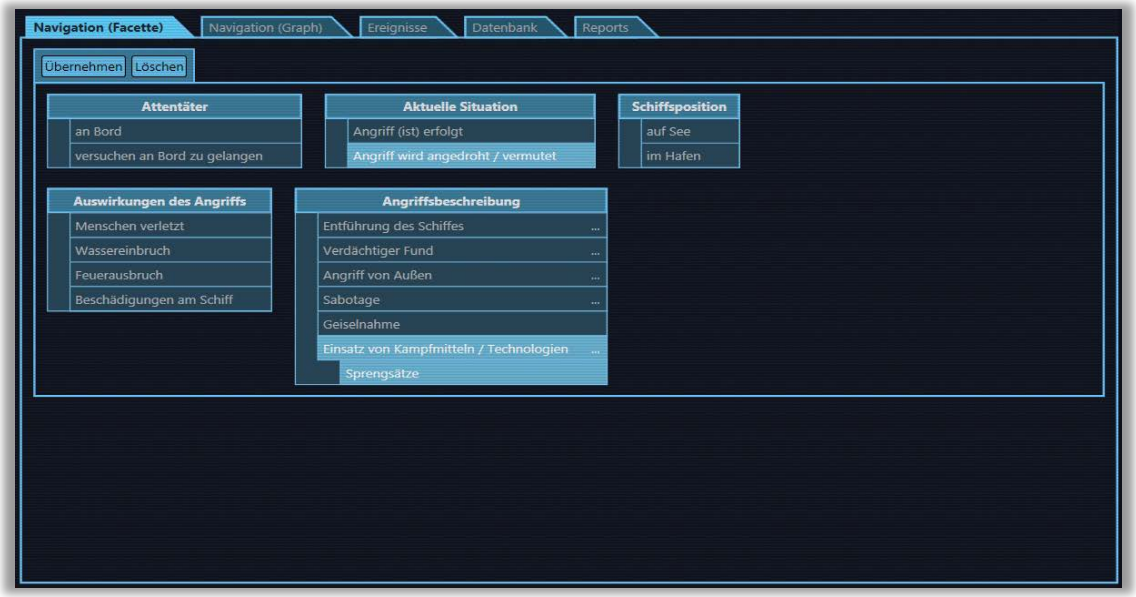

Abb. 16: Faceted Search Browser mit ausgewählten Ereignissen.

2.5 Decksansicht

Bisherige Entscheidungsunterstützungssysteme stellen das Schiff mittels 2D Deckspläne dar. Die entstandene Software zeigt, inwieweit eine dreidimensionale Darstellung durch die intuitive, lebensechte Sicht auf das Schiff bessere Erfolge in der Informationsaufnahme durch den Menschen erzielt. Dabei sind zwei Punkte interessant: Die virtuelle Kameraposition und die Aufteilung des Schiffskörpers.

Die Anzahl der Rotationsfreiheitsgrade im n-dimensionalen Raum ist n*(n-1)/2. Für den zweidimensionalen Raum existiert dementsprechend ein Rotationsfreiheitsgrad für die virtuelle Kamera. Bei der Ansicht der 2D Deckspläne in bisherigen EUS wird die Ansicht so gedreht, dass das die Längsachse des Schiffs waagerecht auf dem Bildschirm liegt. In der dreidimensionalen Darstellung existieren drei Rotationsfreiheitsgrade. Demnach gibt es im 3D Raum mehr Möglichkeiten für die Rotation einer virtuellen Kamera. So können unnatürliche Ansichten erzeugt werden, die der Informationsaufnahme hinderlich sind. Wenn das Programm dem Nutzer zu viel Freiheit über die Kameraführung gibt, wird er möglicherweise in Situationen kommen, in denen oben und unten vertauscht sind oder das Schiff gänzlich aus dem Sichtfeld verschwunden ist. Darum ist die Wahl der richtigen Kamerasteuerung maßgeblich für die Interaktion mit der 3D Szene.

Für den EUS-Demonstrator wurden mehrere Steuerungsalternativen implementiert, um das subjektive Empfinden der Benutzer zu analysieren. Die Modelle reichen von festen Kameraeinstellungen über verschiedene Restriktionen bis hin zur uneingeschränkt, frei beweglichen Kamera mit drei Translations- und drei Rotationsfreiheitsgraden. Für die Umsetzung einer festen Kameraeinstellung ist es denkbar, grobe Gebiete des Schiffes festzulegen (z.B. Vorne, Mitte, Hinten) und die Kamera je nach Geschehen und Benutzerwunsch auf eines der Gebiete zu zentrieren. Etwas mehr Kontrolle über die Kamera erlangt der Benutzer mit der sogenannten Trackball-Steuerung. Hierbei wird ein Punkt auf dem Schiff festgelegt, um den die Kamera frei gedreht werden kann. Das Festlegen des Punktes kann beispielsweise durch

Mausklick erfolgen. Durch weitere Restriktionen können unnatürliche Ansichten verhindert werden. In allen Ansichten mit festem Blickpunkt kann eine Vergrößerungsfunktion eingebaut werden, so dass der Betrachter näher an das Schiff „heranfahren“ kann.

Es wurden verschiedene Ansätze zur Betrachtung einzelner Decks und deren Räumlichkeiten implementiert. Dabei stellte sich heraus, dass eine geeignete Variante das „Schubladensystem“ ist. Hierbei wird das gewünschte Deck mit einem Mausklick seitlich aus dem Schiff (wie eine Schublade aus einem Schrank) herausgezogen. Das Deck befindet sich aus Betrachtersicht vor dem Rest des Schiffes, wodurch die Gesamtsicht, im Gegensatz zur zweidimensionalen Darstellung einzelner Decks, erhalten bleibt. Ereignisse werden in Form von Symbolen oder realitätsnahen Effekten an der entsprechenden Position angezeigt. Wenn ein Deck „geschlossen“ ist (wie eine geschlossene Schublade), werden die Ereignisse dieses Decks seitlich an der Außenbordwand angezeigt.

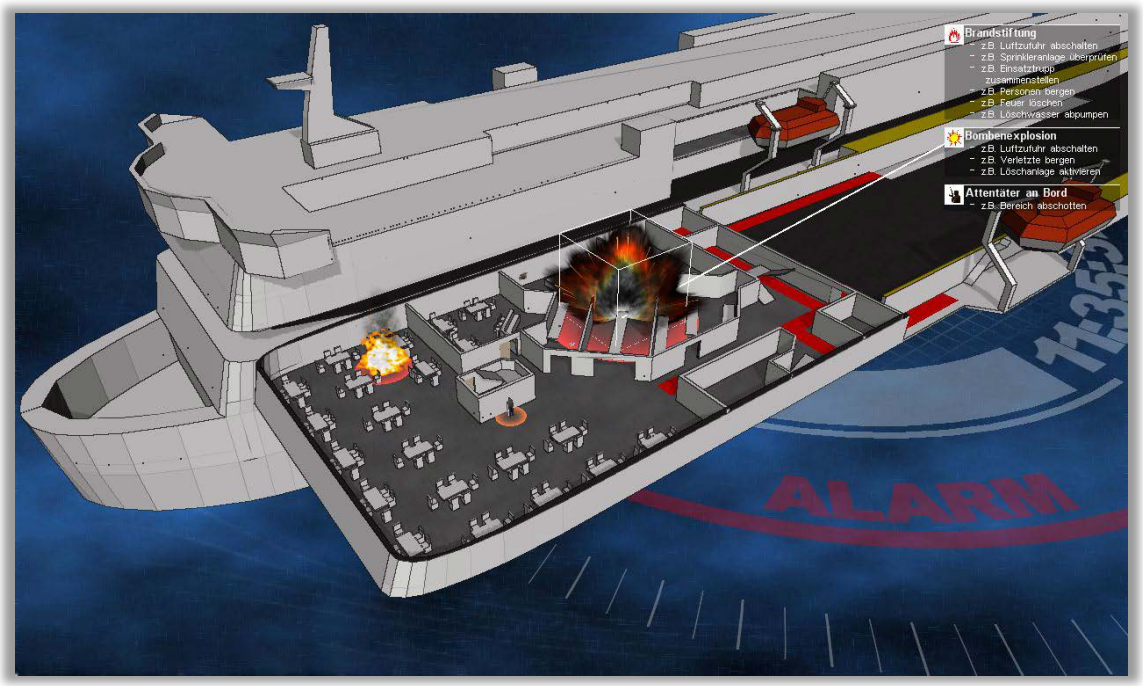

Abb. 17: Cafeteriadeck der Fähre Mecklenburg-Vorpommern im EUS-Demonstrator mit einer Gefahrensituation bestehend aus drei Ereignissen.

Für den Szenenhintergrund wurden zwei Varianten implementiert. Die realitätsnahe Variante zeigt eine Wasseroberfläche und einen Himmel. Die abstrakte Variante bietet die Möglichkeit, weitere Informationen darzustellen (Hier die Uhrzeit und der Alarmstatus).

Im Vordergrund befindet sich eine Auflistung der aktiven Ereignisse und Maßnahmen. Eine optische Verbindung zwischen einem Ereignis in der Auflistung und auf dem Schiffsdeck lässt sich durch Mauseingaben herstellen, so dass die Ereignisse schneller lokalisiert werden können. Die Positionierung der Ereignisse erfolgt durch drag’n’drop.

Optische Verbesserungen wie Schatten und Kantenhervorhebung verstärken den dreidimensionalen Eindruck. Fluchtwege werden in einer experimentellen Fassung als farbige Flächen dargestellt. Zur Demonstration werden Teile der Fluchtwege als gesperrt markiert, sobald sich ein Objekt darauf befindet.

2.6 Integration in bestehende Systeme

Das Safety-EUS MADRAS vom Projektpartner Marsig basiert ebenfalls auf Microsoft WPF und ist daher technisch kompatibel zu den Komponenten des EUS-Demonstrators. Nach Absprache bei einem Arbeitstreffen wurden Teile des EUS-Demonstrators in MADRAS prototypenhaft integriert bzw. dort nachgebildet.