Technische Universität Braunschweig

NeXT Generation Transport Tycoon

Team 0

Software-Entwicklungspraktikum (SEP)
Sommersemester 2013

Pflichtenheft

Auftraggeber:
Technische Universität Braunschweig
Institut für Programmierung und Reaktive Systeme
Prof. Dr. Ursula Goltz
Mühlenpfordtstr. 23
38106 Braunschweig

Betreuer: Benjamin Mensing

Auftragnehmer:

Name	E-Mail-Adresse
Dennis Stelter	d.stelter@tu-bs.de
Henrik Lange	henrik.lange@tu-bs.de
Jochen Steiner	jochen.steiner@tu-bs.de
Markus-Björn Meißner	m-b.meissner@tu-bs.de
Patricia-Tatjana Kasulke	p.kasulke@tu-bs.de
Sebastian Eilf	s.eilf@tu-bs.de
Tessa Fabian	tessa.fabian@tu-bs.de

Braunschweig, 17. April 2013

Versionsübersicht

Version	Datum	Autor	Status	Kommentar
0.0	07.04.2013	Henrik Lange	i.B.	Titelseite geändert
0.1	11.04.2013	Henrik Lange	i.B.	Produktübersicht hinzugefügt
0.2	11.04.2013	Markus-Björn Meißner	i.B.	Produktfunktionen hinzugefügt
0.3	11.04.2013	Jochen Steiner	i.B.	Benutzeroberfläche
0.4	11.04.2013	Tessa Fabian	i.B.	Produktdaten hinzugefügt
0.5	11.04.2013	Dennis Stelter	i.B.	Nichtfunktionale Anforderungen hinzugefügt
0.6	11.04.2013	Sebastian Eilf	i.B.	Technische Produktumgebung hinzugefügt
0.7	11.04.2013	Patricia-Tatjana Kasulke	i.B.	Produkteinsatz hinzugefügt
0.8	11.04.2013	Patricia-Tatjana Kasulke, Tessa Fabian, Henrik Lange	i.B.	Zielbestimmung hinzugefügt
0.9	12.04.2013	Tessa Fabian, Henrik Lange	i.B.	Glossar hinzugefügt
0.10	15.04.2013	Tessa Fabian, Dennis Stelter, Henrik Lange	i.B.	Korrektur von Rechtschreibfehlern
0.11	15.04.2013	Henrik Lange	i.B.	Korrektur von Diagrammen
0.12	16.04.2013	Markus-Björn Meißner	i.B.	Produktfunktionen ergänzt
0.13	17.04.2013	Patricia-Tatjana Kasulke	i.B.	Einleitungstexte ergänzt und Zielbestimmung korrigiert
1.0	17.04.2013	Henrik Lange	i.B.	Abgabe ISF

i.B.: in Bearbeitung

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Zielbestimmung

Ziel des Projektes ist es das bekannte und beliebte Spiel Transport Tycoon aus dem Jahr 1994 mit Hilfe von Lego Mindstorms NXT Robotern umzusetzen.
Die Roboter bewegen sich auf einem festgelegten Straßennetz, das durch schwarzes Klebeband auf dem Boden eines Raumes markiert wird, dem sie mit der Hilfe von Sensoren folgen.
Gewünscht ist ein Konkurrenzverhalten, das sich durch Überbieten und intelligentes Wirtschaften ausdrückt. Zudem sollen eine künstliche Intelligenz besitzen und selbständig entscheiden können. Eine Flotte von Robotern repräsentiert ein Transportunternehmen. Diese gehört zu einem Spieler und soll Transportaufträge ausführen.
Der Warentransport wird dabei rein virtuell durch die Roboter abgebildet. Die Produktion der Industrie soll dabei auch auf die Menge der zugeführten Waren mit zeitlicher Verzögerung oder Beschleunigung reagieren.
Auch die Industrien sind Simulationen, deren Warenein- und ausgang, Kapazitäten und wirtschaftliche Daten auf einer grafischen Oberfläche angezeigt werden sollen. Über diese Oberfläche soll ebenfalls die Bedienung des Spiels möglich sein. Zum Beispiel das Ändern der Ressourcen, beispielsweise der Verfügbarkeit, der Anzahl der Industrien oder der Produkte.
Auf dem Straßennetz gibt es festgelegte Punkte, zum Beispiel am Ende einer Straße, die Industriestandorte darstellen. Sie dienen somit als Start und Ziel der Transportaufträge und besitzen die Marktkräfte Angebot und Nachfrage, die der Preisbildung dienen.
Gewünscht ist zudem noch eine Implementierung eines Computergegners, der verschiedene Strategien anwendet.

1.1 Musskriterien

Folgende Kriterien müssen umgesetzt werden:

$\langle RM1 \rangle$ Der Roboter erkennt den Straßenverlauf in Form von schwarzen Linien und folgt ihnen
$\langle RM2 \rangle$ Der Roboter führt angenommene Aufträge selbständig durch.
$\langle RM3 \rangle$ Der Roboter kann zwischen zwei Punkten den Weg selbstständig finden.
$\langle RM4 \rangle$ Der Roboter muss Aufträge ersteigern können.
$\langle RM5 \rangle$ Der Roboter muss mit dem Server kommunizieren können.
$\langle RM6 \rangle$ Der Roboter hat eine maximale Ladekapazität.
$\langle RM7 \rangle$ Der Roboter muss Deadlock-Situationen erkennen und vermeiden können.

⟨$RM8$⟩ Der Server muss das Wegenetz verwalten.
⟨$RM9$⟩ Der Server muss Aufträge verwalten.
⟨$RM10$⟩ Der Server muss den Roboter hinsichtlich der Kartendaten initialisieren.
⟨$RM11$⟩ Der Server muss die Preisbildung verwalten.
⟨$RM12$⟩ Der Server verwaltet die Kommunikation zwischen Benutzeroberfläche und Roboter.
⟨$RM13$⟩ Der Server muss mit mehreren Robotern gleichzeitig kommunizieren.
⟨$RM14$⟩ Der Server muss den Standort der Roboter kennen.
⟨$RM15$⟩ Der Server muss Auftragsauktionen anbieten.
⟨$RM16$⟩ Der Server muss Geschäftsstatistiken für die Roboter führen.
⟨$RM17$⟩ Der Server muss es ermöglichen, dass Ressourcen und Produkte ergänzt werden können.
⟨$RM18$⟩ Die Spieleroberfläche muss das Wegenetz visualisieren.
⟨$RM19$⟩ Die Spieleroberfläche muss eine Übersicht über die Aufträge bieten.
⟨$RM20$⟩ Die Spieleroberfläche muss eine Statistik über die Roboter anzeigen.
⟨$RM21$⟩ Die Spieleroberfläche muss die Teilnahme an Auftragsauktionen ermöglichen.
⟨$RM22$⟩ Die Visualisierung des Wegenetz muss den Standort und Status von Robotern und Industriestandorten enthalten.
⟨$RM23$⟩ Die Karte des Wegenetzes kann leicht vom Benutzer an die örtlichen Bedingungen (aufgeklebtes Straßennetz) angepasst werden.
⟨$RM24$⟩ Die Spieloberfläche muss Details (vorhandene Waren zum Transport, gelagerte Waren zur Weiterverarbeitung) einzelner Industrien anzeigen.
⟨$RM25$⟩ Die Anordnung und der Produktionszyklus von Industrien soll leicht änderbar bzw.anpassbar sein.

1.2 Sollkriterien

Folgende Kriterien sollten umgesetzt werden:

⟨$RS1$⟩ Der Roboter verwaltet seinen (virtuellen) Benzinverbrauch.
⟨$RS2$⟩ Der Roboter zeigt eine Statusmeldung mit Transportinformationen an.
⟨$RS3$⟩ Der Roboter kann eigenständig lukrative Aufträge wählen.
⟨$RS4$⟩ Der Server soll eine Administrator-Oberfläche haben.
⟨$RS5$⟩ Der Server soll Benutzeraufträge annehmen können.
⟨$RS6$⟩ Der Spieler soll die vom Roboter vorgeschlagene Route verändern können.

1.3 Kannkriterien

Die nachfolgenden Kriterien können umgesetzt werden:

⟨$RC1$⟩ Der Roboter soll die Verderblichkeit der Waren berücksichtigen.
⟨$RC2$⟩ Der Server kann eine Kalibrierung der Roboter vornehmen.
⟨$RC3$⟩ Die Spieleroberfläche muss von einem anderen PC aus aufrufbar sein.

1.4 Abgrenzungskriterien

Diese Ziele sollen bewusst nicht erreicht werden:

⟨$RW1$⟩ Nach dem Spielstart darf das Wegenetz nicht geändert werden können.
⟨$RW2$⟩ Die Roboter transportieren Waren nicht physisch.
⟨$RW3$⟩ Die grafische Darstellung erfolgt nicht in Echtzeit.
⟨$RW4$⟩ Die Karte wird nicht dreidimensional visualisiert.
⟨$RW5$⟩ Zur Spielzeit dürfen die Eigenschaften von Ressourcen und Produkte nicht verändert werden.

2 Produkteinsatz

In diesem Abschnitt wird zunächst auf den Anwendungsbereich eingegangen. Es wird beschrieben, welchem Zweck das Produkt dient und welche verschiedenen Anwendungsbereiche vorhanden sind. Danach wird erläutert, welche Zielgruppe das Produkt ansprechen soll, sowie welche Qualifikationen diese mitbringen sollte. Zum Schluss werden die Betriebsumgebung und Betriebsvoraussetzungen im Abschnitt Betriebsbedingungen dargelegt.

2.1 Zielgruppen

Die Zielgruppe dieses Produktes sind Benutzer, die Freude an Wirtschaftssimulationen haben, oder Forscher, die verschiedene wirtschaftliche Strategien untersuchen wollen.

2.2 Betriebsbedingungen

Da es sich bei unserem Produkt um ein visualisiertes Spiel handelt, wird ein PC benötigt auf dem das Programm installiert wird. Die ebenfalls notwendigen Lego MindStorms NXT 2.0 Roboter bewegen sich auf einem durch schwarzes Klebeband dargestelltes Straßennetz, dessen Kreuzungen im rechten Winkel aufeinandertreffen.
Die Kommunikation zwischen den Robotern und dem PC findet über Bluetooth statt. Daher muss dieses auf den verwendeten Rechnern vorhanden und während des Spiels aktiviert sein. Die Roboter sollten vor dem Spielstart mit vollgeladenen Akkus ausgestattet werden, um das Spiel nicht unterbrechen oder beenden zu müssen.

3 Produktübersicht

Das Produkt wird von verschiedenen Aktoren (Administrator, menschlicher Spieler und Roboter) verwendet, die unterschiedliche Berechtigungen besitzen. Die verschiedenen Rollen werden mit Hilfe eines Use-Case-Diagramms, welches die verschiedenen Akteure, mit ihrem externen Verhalten, ihren Beziehungen zueinander aus Sicht des Nutzer darstellt, erläutert.

3.1 Use-Case-Diagramm

In dem Use-Case-Diagramm (Abb. 3.1) kann man sehen welche unterschiedliche Aufgaben die jeweiligen Aktoren haben. So erkennt man, dass der Admin der einzige ist, der Einstellungen importieren kann, darunter fällt unter anderem der Graph. Zudem darf er neben den Aufträgen der einzelnen Industrien weiter Aufträge einstellen. Wird ein Auftrag ausgeschrieben, so findet eine Auktion statt, wo der Bieter mit dem niedrigsten Gebot gewinnt, da die Industrie ihre Kosten möglich gering halten wollen, der genauere Ablauf wird im Kapitel 3.2 erläutert. Wenn der Admin alle nötigen Einstellungen getätigt hat, kann er das Spiel starten und somit die Einstellungen an die Roboter übertragen.

Während des Spiels übermittelt der Roboter immer wieder seine Position und seine nächste zu besuchende Kante. Erkennt der Server hierbei ein Problem, wenn z.B. ein anderer Roboter sich auf dieser Kante befindet, so wird dies dem Roboter gemeldet und er berechnet sich – wenn möglich eine Alternativroute.

Wenn einer der Spieler ein menschlicher Spieler ist, so hat er die Möglichkeit, die vorgeschlagene Route nach seinen Wünschen zu verändern. Der Spieler kann aber nicht nur die Routen verändern, er übernimmt ganz oder teilweise Aufgaben, die sonst die KI auf dem Roboter durchführen würde.

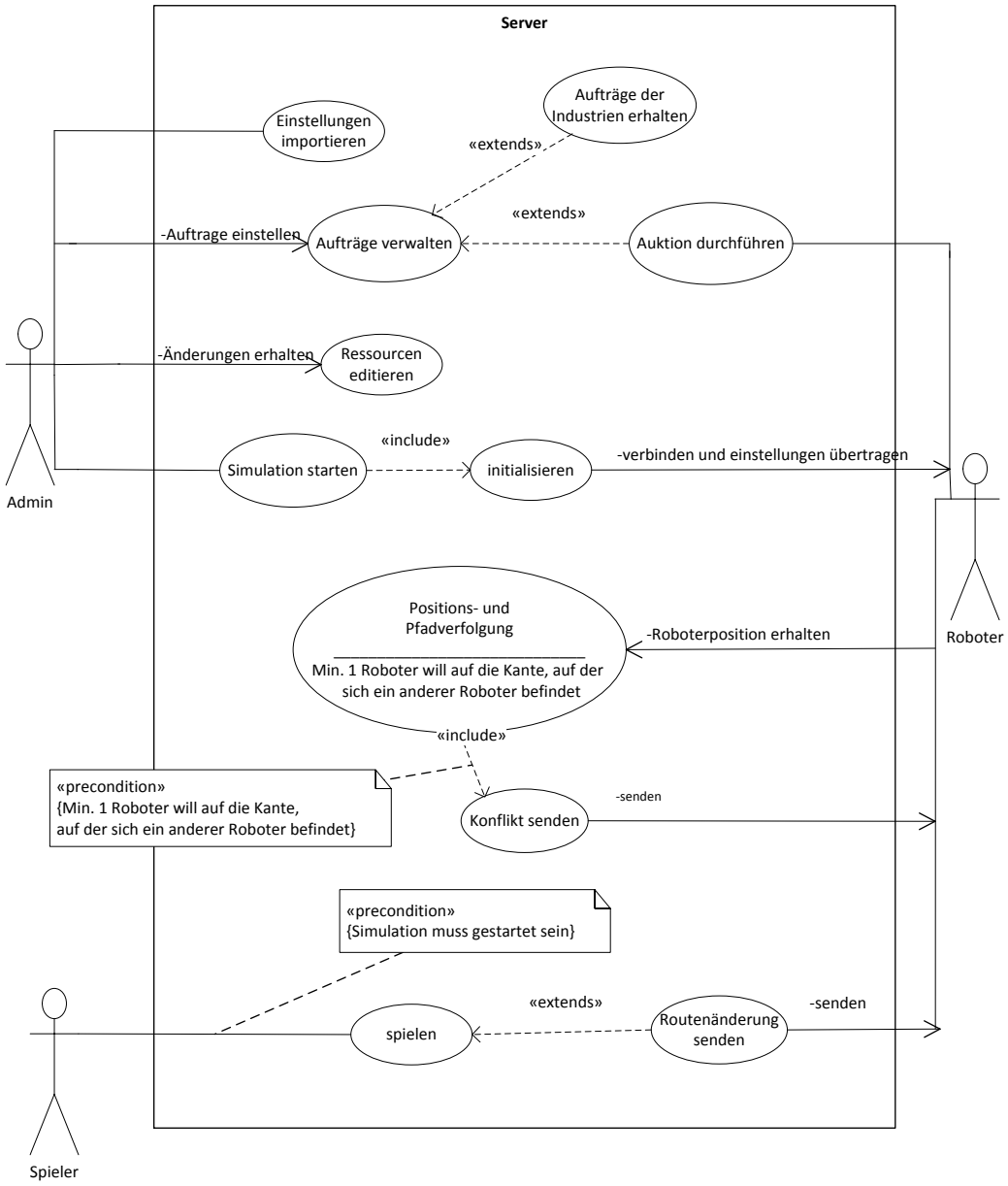

Abbildung 3.1: Use-Case Diagramm

3.2 Aktivitätsdiagramm

Mit Hilfe des Aktivitäsdiagramms wird, der im Use-Case-Diagram erläuterte Anwendungsfall genauer beschrieben. Dabei wurde die zeitliche Abfolge der Aktionen miteinbezogen.

In dem Aktivitätsdiagram(Abb. 3.2) sehen wir, das ein Admin einen Auftrag erstellt. Will er diesen zur Auktion freigeben, so muss vorher überprüft werden, ob es möglich ist, eine Nachricht an die Roboter zu versenden. Ist dies aus verschiedenen Gründen nicht möglich, so kann der Vorgang nicht fortgeführt werden.
Wenn das Angebot erfolgreich an alle Roboter versandt wurde, so gibt jeder Roboter ein Gebot ab. Dieses Gebot berechnen die Roboter anhand verschiedenster Kriterien, so dass das Ergebnis rentabel ist.
Der Server überprüft daraufhin, wer das niedrigste Gebot abgegeben hat und benachrichtigt dann die Roboter über den Ausgang. Hat ein Roboter den Auftrag erhalten, muss er diesen zu seiner Route hinzufügen.

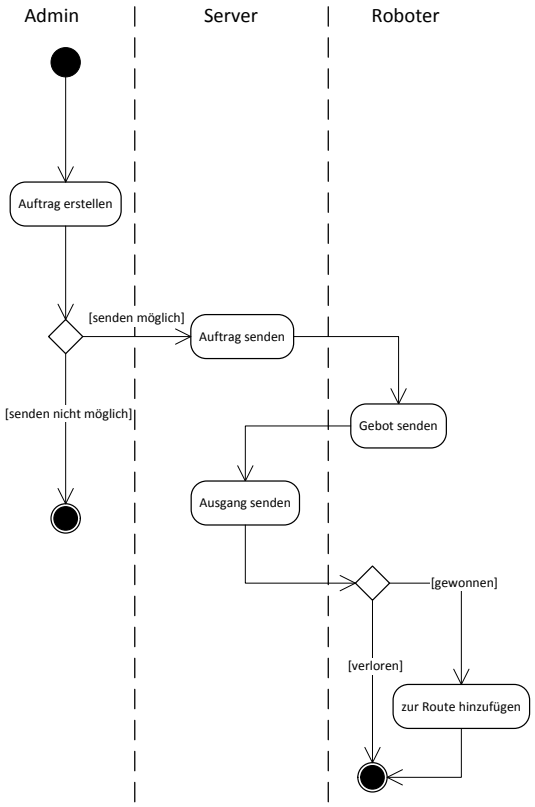

Abbildung 3.2: Auftrag

4 Produktfunktionen

Bluetooth-Datenverbindung $\langle F10 \rangle$

Geschäftsprozess: Datenübertragung zwischen Roboter und Server.

Anforderung: $\langle RM5 \rangle$, $\langle RM13 \rangle$

Ziel: Datenübertragung zwischen Robotern und Server zur Koordination des Spielablaufs.

Vorbedingung: Roboter oder Server haben Daten zum Senden.

Nachbedingung Erfolg: Erfolgreiche Datenübertragung.

Nachbedingung Fehlschlag: Fehlerhafte Datenübertragung.

Akteure: Roboter, Server.

Auslösendes Ereignis: Umgebungsdaten müssen vor Spielbeginn auf teilnehmende Roboter übertragen werden. Server will Aufträge bekanntmachen. Während des Spieles fragen Roboter beispielsweise um Freigabe des nächsten zu befahrenden Streckenabschnittes.

Beschreibung:

1. Zur drahtlosen Kommunikation wird Bluetooth verwendet.
2. Der Server ist in der Lage mit mehreren Robotern gleichzeitig zu kommunizieren.

Datenverwaltung $\langle F20 \rangle$

Geschäftsprozess: Initialisierung und Datenverwaltung des Servers.

Anforderung: $\langle RM8 \rangle$, $\langle RM9 \rangle$, $\langle RM11 \rangle$, $\langle RM12 \rangle$, $\langle RM14 \rangle$, $\langle RM17 \rangle$, $\langle RM23 \rangle$, $\langle RM25 \rangle$, $\langle RS4 \rangle$

Ziel: Der Benutzer konfiguriert den Server. Der Server speichert die Daten der durchgeführten Aufträge der Roboter und berechnet den jeweils erzielten Gewinn und Statistiken.

Vorbedingung: Eingabe bzw. Laden der Daten auf den Server durch den Benutzer über die Benutzeroberfläche.

Nachbedingung Erfolg: Die gewünschten Daten werden vom Server korrekt verarbeitet und der GUI zur Verfügung gestellt.

Nachbedingung Fehlschlag: Daten sind nicht korrekt oder nicht vollständig.

Akteure: Server, Benutzer.

Auslösendes Ereignis: Ein neues Spiel soll durch den Benutzer konfiguriert werden.

Beschreibung:

1. Dateneingabe am Server über die Benutzer GUI.
2. Das Eingeben von Wegenetz und Ressourcen sowie Produktionszyklen ist notwendig und nur vor Spielbeginn möglich.
3. Die Startpositionen der Roboter werden ebenfalls vor dem Spiel festgelegt.
4. Während des Spiels können eigene Aufträge durch den Benutzer erstellt werden.
5. Während des Spiels errechnet der Server die Preisbildung aus Angebot und Nachfrage der Waren.
6. Gewinne und Statistiken der Roboter werden vom Server aus den Daten des laufenden Spiels berechnet.

Initialisierung der Roboter $\langle F30 \rangle$

Geschäftsprozess: Server initialisiert Roboter.

Anforderung: $\langle RM10 \rangle$

Ziel: Roboter werden zu Spielbeginn initialisiert, damit sie am Spiel teilnehmen können.

Vorbedingung: Server kennt das Wegenetz.

Nachbedingung Erfolg: Der Roboter ist spielbereit und wartet auf Auftragsübermittlung durch den Server.

Nachbedingung Fehlschlag: Ohne korrekte Initialisierungsdaten kann der Roboter nicht am Spiel teilnehmen.

Akteure: Roboter, Server.

Auslösendes Ereignis: Roboter soll am Spiel teilnehmen.

Beschreibung:

1. An die verfügbaren Roboter werden zu Spielbeginn die Daten des Wegenetzes übermittelt.
2. Diese Daten sind erforderlich, damit die KI des Roboters Berechnungen durchführen kann.
3. Der Roboter bestätigt dem Server das Erhalten der Daten und signalisiert damit die Teilnahme am Spiel.

Straßenerkennung durch Roboter $\langle F40 \rangle$

Geschäftsprozess: Roboter folgt dem Straßenverlauf.

Anforderung: $\langle RM1 \rangle$

Ziel: Der Roboter folgt dem Straßenverlauf. Er erkennt gerade Strecken, Kurven, Kreuzungen und Industriestandorte.

Vorbedingung: Der Roboter befindet sich auf dem Straßennetz. Seine Sensorik ist auf die schwarze Linienmarkierung ausgerichtet.

Nachbedingung Erfolg: Der Roboter ist dem Verlauf der Straße gefolgt.

Nachbedingung Fehlschlag: Die Sensorik des Roboters ist nicht mehr auf die schwarze Linienmarkierung ausgerichtet. Der Roboter hat die Orientierung verloren.

Akteure: Roboter.

Auslösendes Ereignis: Der Roboter bearbeitet einen Transportauftrag und möchte sich zum Erreichen seines Ziels auf dem Wegenetz fortbewegen.

Beschreibung:

1. Eine schwarze Linienmarkierung stellt die Straßenmitte dar.
2. Der Roboter unterscheidet helle und dunkle Farben mittels Sensorik.
3. In Bewegung muss der Roboter Farbabweichungen erkennen und seinen Kurs entsprechend korrigieren.
4. Das Erreichen oder Verlassen von Kreuzungen wird durch eine zusätzliche schwarze Querlinie erkannt.
5. Industriestandorte werden durch eine schwarze Querlinie an einem Straßenende gekennzeichnet.
6. Der Roboter kennt spezielle Bewegungsabläufe zum Wenden und zum Befahren von rechtwinkligen Kreuzungen.
7. An Straßenenden muss der Roboter Wendemanöver vollziehen können.

Transportvolumen der Roboter $\langle F50 \rangle$

Geschäftsprozess: Roboter haben eine maximale Ladekapazität.

Anforderung: $\langle RM6 \rangle$, $\langle RS2 \rangle$

Ziel: Roboter können nicht beliebig viele Waren zur gleichen Zeit transportieren.

Vorbedingung: Roboter wurde als Transportmittel einer bestimmten Größe geplant.

Nachbedingung Erfolg: Roboter berücksichtigt seine maximale Ladekapazität bei der Auftragsannahme.

Nachbedingung Fehlschlag: Roboter transportiert zu viele Waren auf einmal.

Akteure: Roboter.

Auslösendes Ereignis: Roboter KI berechnet Lukrativität eines Auftrags.

Beschreibung:

1. Es existiert eine zweckbestimmte Begrenzung der Ladekapazität eines Roboters, um beispielsweise das Aufteilen von Aufträgen auf mehrere Roboter zu forcieren.
2. Roboter verfügen über ein Display, das Transportinformationen darstellt.
3. Roboter achten darauf, welche Waren zusammen transportiert werden dürfen.
4. Roboter, die sich überladen haben, sollen aus dem Spiel ausscheiden.

Treibstoffverbrauch der Roboter $\langle F60 \rangle$

Geschäftsprozess: Roboter haben einen virtuellen Benzinverbrauch.

Anforderung: $\langle RS1 \rangle$

Ziel: Roboter brauchen zur Bearbeitung ihrer Transporte einen virtuellen Treibstoff. Roboter dürfen nicht weiterfahren, falls sie keinen Treibstoff mehr haben.

Vorbedingung: Der Roboter kennt die Restmenge seines Treibstoffvorrats.

Nachbedingung Erfolg: Routen werden unter Berücksichtigung der Treibstoffmenge geplant. Ohne Treibstoff, kann ein Roboter keine Aufträge mehr bearbeiten.

Nachbedingung Fehlschlag: Der Roboter fährt weiter, obwohl er keinen Treibstoff mehr hat.

Akteure: Roboter.

Auslösendes Ereignis: Von Spielbeginn an verbraucht der Roboter seinen Treibstoff.

Beschreibung:

1. Der Treibstoffverbrauch ist zeitabhängig.
2. Mit erfolgreich abgeschlossenen Aufträgen wird ein Roboter neu betankt.

Streckenfreigabe durch Server $\langle F70 \rangle$

Geschäftsprozess: Roboter bittet um Freigabe eines Streckenabschnittes.

Anforderung: $\langle RM3 \rangle$

Ziel: Server erteilt einem Roboter die Erlaubnis einen bestimmten Streckenabschnitt befahren zu dürfen.

Vorbedingung: Der Roboter bearbeitet einen Auftrag und hat zuvor eine Route berechnet.

Nachbedingung Erfolg: Der Roboter darf den freigegebenen Streckenabschnitt befahren. Für andere Roboter ist dieser Streckenabschnitt dann gesperrt.

Nachbedingung Fehlschlag: Der Streckenabschnitt wurde nicht freigegeben.

Akteure: Roboter, Server.

Auslösendes Ereignis: Roboter will nächsten Streckenabschnitt seiner Route befahren.

Beschreibung:

1. Mit Erteilung der Freigabe vermerkt der Server für zukünftige Anfragen, dass der Abschnitt belegt ist.
2. Erst wenn ein Roboter signalisiert, dass er einen Abschnitt verlassen hat, vermerkt der Server diesen Abschnitt wieder als frei.

Routenberechnung durch Roboter $\langle F80 \rangle$

Geschäftsprozess: Selbständige Wegewahl des Roboters.

Anforderung: $\langle RM2 \rangle$, $\langle RM3 \rangle$, $\langle RS6 \rangle$

Ziel: Der Roboter soll selbständig eine geeignete Route zum Ziel seines Auftrages berechnen.

Vorbedingung: Der Roboter kennt das Straßennetz und die Streckenkosten und hat einen Auftrag in Bearbeitung.

Nachbedingung Erfolg: Der Roboter verfolgt die geplante Route bis zum Ziel.

Nachbedingung Fehlschlag: Eine Freigabe wurde nicht erteilt. Der Roboter muss eine alternative Route berechnen.

Akteure: Roboter.

Auslösendes Ereignis: Erstmalig bei Auftragsannahme und im späteren Verlauf bei einer Ablehnung einer Freigabeanfrage für einen Streckenabschnitt.

Beschreibung:

1. Der Roboter berechnet eine Route anhand der gespeicherten Wegenetzdaten.
2. Die Routenberechnung ist abhängig von der gewählten KI des Roboters.
3. Der Roboter folgt der berechneten Route.

Alternativen:

3a. Menschliche Spieler können die von ihren Robotern vorgeschlagenen Routen ablehnen, um diesen eigene Routen vorzugeben.

Spielstart durch Benutzer $\langle F90 \rangle$

Geschäftsprozess: Ein neues Spiel wird gestartet.

Anforderung: $\langle RS4 \rangle$

Ziel: Das Spiel läuft. Der Server beginnt mit der Auftragsversteigerung und die Roboter bzw. Spieler können ihr erstes Gebot abgeben.

Vorbedingung: Die Initialisierung des Server ist abgeschlossen. Die Roboter stehen spielbereit auf den gewünschten Positionen des Spielfeldes.

Nachbedingung Erfolg: Das Spiel läuft.

Nachbedingung Fehlschlag: Das Spiel konnte nicht gestartet werden.

Akteure: Benutzer, Server, Roboter.

Auslösendes Ereignis: Ein Spiel soll gestartet werden.

Beschreibung:

1. Der Server wird initialisiert.
2. Die Roboter werden auf dem Spielfeld positioniert.
3. Der Benutzer betätigt den Start-Knopf auf der Benutzeroberfläche.
4. Nach erfolgreicher Initialisierung der Roboter (Funktion $\langle F10 \rangle$) und Vollständigkeitsprüfung durch den Server, wird das Spiel gestartet.

Erweiterung:

4a. Bei menschlichen Spielern wird der Spielbeginn in der Spieler GUI angezeigt.

Spieldarstellung der Spieleroberfläche $\langle F100 \rangle$

Geschäftsprozess: Die Spieleroberfläche zeigt Wegenetz mit Standorten von Industrien und Robotern an.

Anforderung: $\langle RM18 \rangle$, $\langle RM22 \rangle$, $\langle RM24 \rangle$

Ziel: Der Spieler soll das Spiel über eine graphische Visualisierung am Computer mitverfolgen können.

Vorbedingung: Ein menschlicher Spieler nimmt am Spiel teil. Das Spiel wurde gestartet.

Nachbedingung Erfolg: Visualisierung des Spielverlaufs auf dem Computer.

Nachbedingung Fehlschlag: Fehlerhafte Visualisierung des Spielverlaufs.

Akteure: Server, Roboter, Spieler.

Auslösendes Ereignis: Mit Spielbeginn ständige Aktualisierung.

Beschreibung:

1. Der Server übermittelt die Daten an das Spielerinterface.
2. Die Darstellung des Wegenetzes erfolgt in Form einer 2D-Ansicht.
3. Belegte und freie Knoten (Kreuzungen) und Kanten (Strecken zwischen den Knoten) werden dargestellt.
4. Die Position der Roboter wird schrittweise anhand der von ihnen belegten Kanten bzw. Knoten dargestellt.

Statistikanzeige der Spieleroberfläche $\langle F110 \rangle$

Geschäftsprozess: Die Spieleroberfläche zeigt laufende Aufträge und Statistiken von Robotern an.

Anforderung: $\langle RM16 \rangle$, $\langle RM19 \rangle$, $\langle RM20 \rangle$, $\langle RM21 \rangle$

Ziel: Der Spieler kann geleistete und laufende Aufträge sowie Gewinne und Statistiken der Roboter in einer Übersicht am Computer mitverfolgen.

Vorbedingung: Der Spieler hat bestenfalls bereits einen Auftrag ersteigert.

Nachbedingung Erfolg: Statistiken werden korrekt berechnet und angezeigt.

Nachbedingung Fehlschlag: Fehlerhafte Anzeige der Statisktiken.

Akteure: Server, Roboter.

Auslösendes Ereignis: Zu Spielbeginn. Dann Aktualisierung bei jeder Auftragsannahme und bei jeder Auftragsbeendigung.

Beschreibung:

1. Aus den auf dem Server vorliegenden Spieldaten werden durchschnittliche Gewinne, gesamter Gewinn oder freie Ladekapazität für jeden Roboter durch den Server berechnet.
2. Die Ergebnisse können vom Spieler über das Spielerinterface abgerufen werden.
3. Über ein Eingabefeld können Spieler Gebote für Auftragsversteigerungen abgeben.

Auktionen für Transportaufträge $\langle F120 \rangle$

Geschäftsprozess: Versteigerung von Aufträgen an Roboter bzw. Spieler.

Anforderung: $\langle RM2 \rangle$, $\langle RM4 \rangle$, $\langle RM15 \rangle$, $\langle RM21 \rangle$, $\langle RS3 \rangle$, $\langle RS5 \rangle$

Ziel: Mit der Zielsetzung eigene Gewinne zu maximieren, konkurrieren die Roboter mittels Geboten um die Auftragserteilung durch den Server.

Vorbedingung: Der Server versendet dasselbe Auftragsangebot an alle teilnehmenden Roboter.

Nachbedingung Erfolg: Server hat Antwort bzw. Gebot von jedem Roboter erhalten. Auftrag wurde an den Roboter mit dem höchsten Gebot vergeben. Dieser Roboter speichert den Auftrag.

Nachbedingung Fehlschlag: Kein Roboter hat einen Auftrag zur Bearbeitung gespeichert.

Akteure: Server, Roboter, Spieler.

Auslösendes Ereignis: Der Server versendet neues Auftragsangebot oder der Benutzer hat manuell einen neuen Auftrag erstellt.

Beschreibung:

1. Der Server versendet Auftragsangebote mittels Broadcast an alle Roboter.
2. Roboter entscheiden abhängig von ihrer KI, welcher Auftrag für sie am lukrativsten ist.

Erweiterung:

1a. Der Benutzer erzeugt während des Spiels neue Aufträge und gibt diese zur Auktion an den Server weiter.

Alternativen:

2a. Der menschliche Spieler legt die Höhe seine Gebote selber fest.

Vermeidung von Deadlocks $\langle F130 \rangle$

Geschäftsprozess: Roboter erkennt Deadlock-Situation.

Anforderung: $\langle RM7 \rangle$

Ziel: Roboter erkennen eine spielblockierende Situation und lösen diese selbständig auf.

Vorbedingung: Mehrere Roboter blockieren sich im nächsten Schritt ihrer Bewegungen gegenseitig.

Nachbedingung Erfolg: Die betreffenden Roboter können ihre Routen zum Zielpunkt weiterverfolgen.

Nachbedingung Fehlschlag: Deadlock. Die betreffenden Roboter blockieren gegenseitig ihre Routen und eine Ausweichroute kann nicht berechnet werden.

Akteure: Roboter.

Auslösendes Ereignis: Die Routen mehrerer Roboter haben einen gleichen kritischen Pfad, d.h einen Streckenabschnitt der nicht umfahren werden kann. Mehrere Roboter wollen nun aus verschiedenen Richtungen einen solchen Streckenabschnitt befahren. An Endpunkten des Wegenetzes kommen solche Pfade beispielsweise vor.

Beschreibung:

1. Ein ungewollter Stillstand der Roboter soll im Spiel vermieden werden.
2. Je kleiner das Spielfeld desto höher die Wahrscheinlichkeit, dass es zu blockierenden Situationen kommt.
3. Je größer die Anzahl der teilnehmenden Roboter desto höher die Wahrscheinlichkeit, dass es zu blockierenden Situationen kommt.
4. Das Blockieren von Wegen als unfaire Spieltaktik soll verhindert werden.

Spieldarstellung über Benutzeroberfläche $\langle F140 \rangle$

Geschäftsprozess: Die Benutzeroberfläche zeigt das Wegenetz, Aufträge und Statistiken an (siehe Abschnitt $\langle UI30 \rangle$).

Anforderung: $\langle RS4 \rangle$, $\langle RM22 \rangle$

Ziel: Der Benutzer kann administrative Aufgaben einfacher erledigen.

Vorbedingung: Das Spiel wurde gestartet.

Nachbedingung Erfolg: Die Benutzeroberfläche stellt alle Informationen korrekt dar.

Nachbedingung Fehlschlag: Es wird keine Benutzeroberfläche angezeigt oder diese wird nicht korrekt dargestellt/aktualisiert.

Akteure: Server, Spieler.

Auslösendes Ereignis: Spielbeginn.

Beschreibung:

1. Aufträge können betrachtet und hinzugefügt werden.
2. Roboter können verwaltet werden.
3. Die Darstellung des Wegenetzes erfolgt in Form einer 2D-Ansicht.
4. Belegte und freie Knoten (Kreuzungen) und Kanten (Strecken zwischen den Knoten) werden dargestellt.
5. Die Position der Roboter wird schrittweise anhand der von ihnen belegten Kanten bzw. Knoten dargestellt.

5 Produktdaten

Industriedaten $\langle D10 \rangle$
Daten der Industrien und ihrer Waren:
- Produzierte Ware
- Rohstoffe
- Waren
- Produktionsvorgang (benötigte Rohstoffe, Produktionszeit)
- Bisher erziehlter Gewinn
- Angebot
- Preis
- ID nach dem folgenden Muster: IndustrieN, wobei N eine natürliche Zahl ist
- Kraftstoff (der Kraftstoff, der vor Ort aufgetankt werden kann)
- Startwerte (z.B. Standardaufträge)
- Maximale Lagerkapazität - Standort auf der Karte

Standardaufträge $\langle D20 \rangle$
Standardaufträge, die zum Untersuchen von wirtschaftswissenschaftlichen Zusammenhängen dienen können. Hierbei handelt es sich nicht um Aufträge, die während einer Spielsitzung generiert werden, sondern um solche, die mehrfach und in mehreren Spielsitzungen zu Forschungszwecken eingesetzt werden können.
- Nummer
- Inhalt
- Preis
- Auftraggeber

Kartendaten $\langle D30 \rangle$
Die Karte wird als Graph betrachtet, der vor Spielbeginn erweitert werden kann:
- Knoten (repräsentieren Industriestandorte und Kreuzungen)
- Kanten (repräsentieren Straßen)

6 Nichtfunktionale Anforderungen

Produktqualität	**sehr gut**	**gut**	**normal**	**nicht relevant**
Funktionalität				
Angemessenheit		x		
Richtigkeit	x			
Interoperabilität		x		
Ordnungsmäßgkeit			x	
Sicherheit				
Zuverlässigkeit		x		
Reife			x	
Fehlertoleranz		x		
Wiederherstellbarkeit			x	
Benutzbarkeit				
Verständlichkeit		x		
Erlernbarkeit			x	
Bedienbarkeit		x		
Effizienz			x	
Zeitverhalten			x	
Verbrauchsverhalten				x
Änderbarkeit				
Analysierbarkeit			x	
Modifizierbarkeit	x			
Stabilität	x			
Prüfbarkeit		x		
Übertragbarkeit				x
Anpassbarkeit			x	
Installierbarkeit			x	
Konformität				x
Austauschbarkeit				x

- $\langle Q10 \rangle$ Die Funktionen $\langle F10 \rangle$ und $\langle F70 \rangle$ dürfen den Roboter nicht so viel Zeit kosten, dass er seine anderen Aktivitäten unterbrechen muss oder nicht mehr korrekt ausführen kann.

- $\langle Q20 \rangle$ Die Funktion $\langle F20 \rangle$ muss ohne große Verzögerung aktualisiert werden.

- $\langle Q30 \rangle$ Der Roboter muss ständigen Kontakt zum Server aufrechterhalten.

- $\langle Q40 \rangle$ Die Algorithmen auf dem NXT müssen möglichst effizient sein, weil die zugrundeliegende Hardware auf den Robotern nur begrenzt leistungsfähig ist.

- $\langle Q50 \rangle$ Das zugrunde liegende Wegenetz muss leicht austauschbar sein.

- $\langle Q60 \rangle$ Die zugrunde liegenden Rohstoffe und Waren müssen leicht austauschbar sein. Darüberhinaus muss man auf einfache Weise neue Industrien hinzufügen können. Anfangs sollen einige Industrien gestellt werden.

- $\langle Q70 \rangle$ Die Roboter müssen (hauptsächlich was die Kollisionsvermeidung betrifft) möglichst fehlerfrei programmiert sein, um den Spielablauf - also besonders andere Roboter - nicht zu behindern.

- $\langle Q80 \rangle$ Die Roboter dürfen das Wegenetz nicht verlassen. Falls dies doch passiert muss der Server entsprechend reagieren.

- $\langle Q90 \rangle$ Die maximale Ladekapazität der Roboter darf sich während einer Spielzeit nicht verändern.

- $\langle Q100 \rangle$ Die Preise von Waren müssen spätestens alle 10 Sekunden aktualisiert werden.

- $\langle Q110 \rangle$ Die Preise von Waren müssen abhängig von Angebot, Nachfrage und Transportdauer sein.

- $\langle Q120 \rangle$ Die KI der Roboter, welche für die Abarbeitungsstrategie der Aufträge zuständig ist, soll leicht austauschbar sein.

7 Benutzeroberfläche

Für das Softwareprodukt werden GUIs zur besseren Bedienung entwickelt.

Rolle	Rechte	Benutzeroberfläche
Admin	$\langle F20 \rangle$, $\langle F30 \rangle$, $\langle F90 \rangle$, $\langle F100 \rangle$, $\langle F110 \rangle$, $\langle F120 \rangle$	Graph, Knöpfe, Listen, Popup für Industriedetails, Popup zum Hinzufügen von Robotern
Spieler	$\langle F100 \rangle$, $\langle F110 \rangle$, $\langle F120 \rangle$	Graph, Knöpfe, Listen, Popup für Industriedetails

Fenster $\langle UI10 \rangle$
Es gibt 2 verschiedene GUI's. Die Erste ist für den Spieler gedacht und die Zweite für den Administrator.

Programmierung $\langle UI20 \rangle$
Beide GUI's sollen mit Java erzeugt werden

Administrator GUI $\langle UI30 \rangle$

- Oben links befindet sich der Knopf zum Starten des Spiels.
- Darunter befindet sich das Wegenetz.
- Unten Links befindet sich ein Knopf, welcher zusätzliche Informationen zu den Industriestandorten bereithält und diese durch betätigen des Knopfes „Industriedetails anzeigen“ in einem neuen Fenster anzeigt $\langle UI50 \rangle$.
- Auf der rechten oberen Seite befindet sich der Abschnitt in dem der Status der Roboter angezeigt wird. Durch den Knopf „Roboter hinzufügen“ gelangt der Administrator in ein neues Fenster, in dem er neue Roboter zum Spiel hinzufügen kann.
- Unterhalb dieses Bereichs befindet sich der Bereich, indem Informationen zu Aufträge angezeigt werden und neue Aufträgen hinzugefügt werden können.

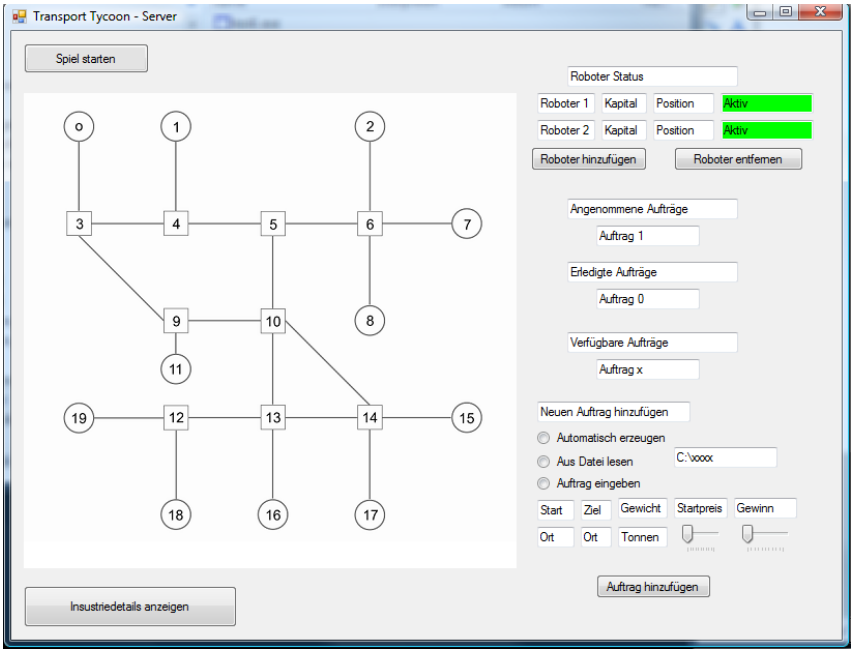

Abbildung 7.1: Administrator GUI $\langle UI30 \rangle$

Roboter hinzufügen $\langle UI40 \rangle$
Durch das Betätigen des Knopfes „Roboter hinzufügen“, auf der Administrator Oberfläche, wird ein Dialogfenster geöffnet. Hier können neue Roboter zum Spiel hinzugefügt und ihre Startwerte verändert werden.

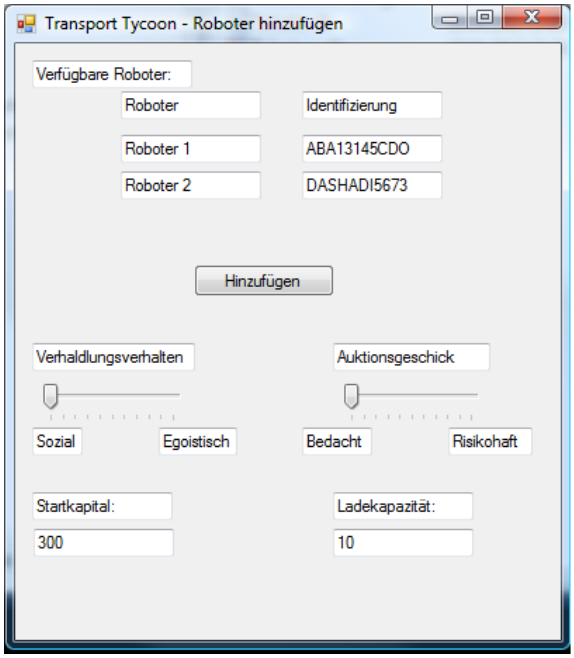

Abbildung 7.2: Roboter hinzufügen $\langle UI40 \rangle$

Verwaltung der Industrien $\langle UI50 \rangle$
Zusätzlich gibt es noch das Dialogfenster, welches über Betätigen des „Industriedetails anzeigen“ Knopfes mehr Informationen zu den Industrien angezeigt:

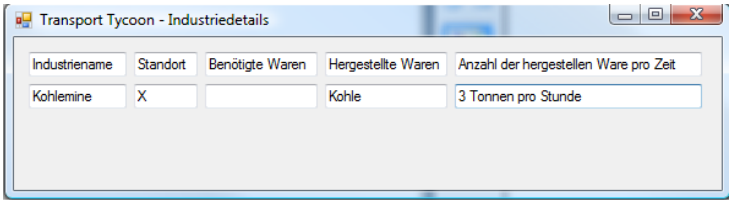

Abbildung 7.3: Verwaltung der Industrien $\langle UI50 \rangle$

Spieler GUI $\langle UI60 \rangle$
Die GUI für den Spieler, ist in 5 Bereiche unterteilt:

- Der Erste befindet sich im linken oberen Teil des Fensters und enthält das Wegenetz.
- Gleich darunter befindet sich der Bereich in dem Auftragsinformationen angezeigt werden. Dort kann ausgewählt werden welche Aufträge angezeigt werden. Beispielsweise Aufträge die zurzeit verfügbar sind. Im selben Bereich kann der Spieler zudem Gebote zu den Aufträgen abgeben.
- Im oberen rechten Teil des Fensters, kann die vom Roboter vorgeschlagene Route durch eine eigene Wunschroute geplant werden.
- Unterhalb davon befindet sich der dritte Bereich, in dem aktuelle Daten und Statistiken, unter anderem findet man hier den Gesamtgewinn des Spielers oder den durchschnittlichen Gewinn.
- Der letzte Bereich ist unten rechts, dort befindet sich der Knopf mir der Aufschrift „Industriedetails anzeigen“$\langle UI50 \rangle$.

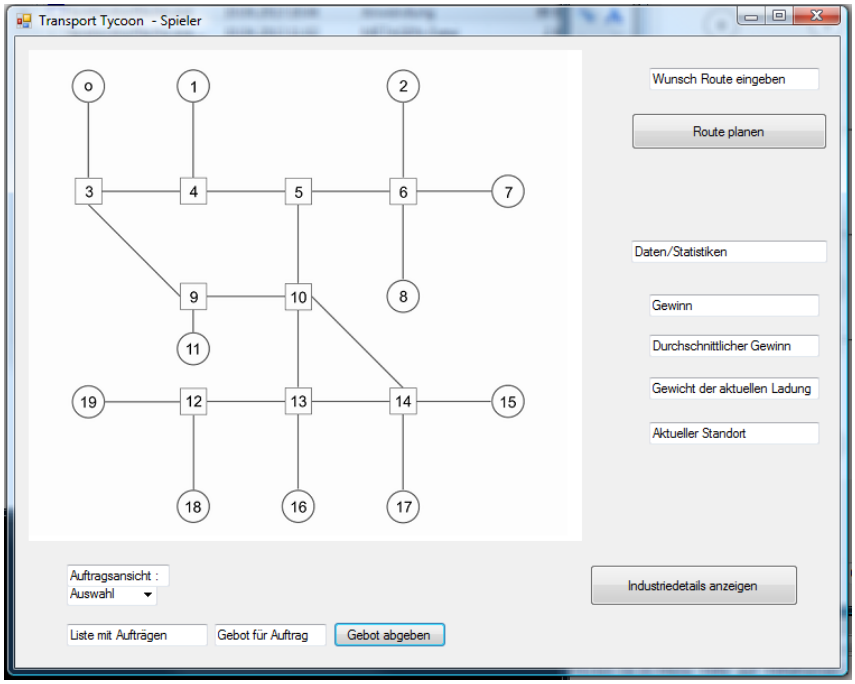

Abbildung 7.4: Spieler GUI $\langle UI60 \rangle$

8 Technische Produktumgebung

Die technische Produktumgebung ist an bestimmte Bedingungen geknüpft, die im Folgenden beschrieben werden.

8.1 Software

Server-Betriebssystem: Windows, Linux (im Allgemeinen alle Betriebssysteme, auf denen ein JDK und ein funktionierender Bluetooth-Treiber installiert sind).
Client-Betriebssystem: Windows, Linux, Mac OS X (im Allgemeinen alle Betriebssysteme, auf denen ein JDK installiert ist).
Roboter-Betriebssystem: leJOS ab Version 0.8.5 beta
Zusätzlich ist eine aktuelle Java-Version ab Version 1.7 auf Server und Client notwendig.

8.2 Hardware

Server: PC, Bluetooth-Modul zur Kommunikation mit dem NXT
Client: PC
Die Programme für Server und Client können auch auf demselben Computer laufen, sodass nur ein System nötig ist, das dann aber beide Anforderungen erfüllen muss. Zusätzlich notwendig sind 1 bis 4 Lego Mindstorms NXT-Roboter mit zugehöriger Sensorik.

8.3 Orgware

Sofern Server und Client nicht auf demselben Computer laufen, müssen beide PCs mithilfe einer TCP/IP-Netzwerkverbindung kommunizieren können.
Weiterhin muss sich das Bluetooth-Modul des Servers zu jedem Zeitpunkt in Funkreichweite des Roboters befinden. Nach Möglichkeit sollten sich keine weiteren Bluetooth- und/oder Funksysteme in Reichweite von NXT und Server befinden, um einen fehlerfreien Ablauf zu gewährleisten. Zudem muss ein „Straßennetz“ auf dem Boden angebracht sein, das aus einer etwa 19 mm breiten, tiefschwarzen Linie auf weißem Grund bestehen und unterbrechungsfrei sein muss, damit der Roboter dem Streckenverlauf problemlos folgen kann. Dieses Straßennetz muss vor Betriebsbeginn auf dem Server abgespeichert werden.

8.4 Produktschnittstellen

Die Kommunikation zwischen Server und Client erfolgt über ein TCP/IP-Netzwerk, sofern beide nicht auf demselben Rechner laufen. Andernfalls ist keine besondere Schnittstelle zur Kommunikation nötig.
Die NXT-Roboter kommunizieren via Bluetooth mit dem Server.

9 Glossar

Auftrag:
Der Auftrag setzt sich aus Auftragnnehmer, Auftraggeber, der zu transportierenden Ware, Start- und Zielpunkt sowie den zu erwartenen Lohn zusammen.

Benutzer:
Menschlicher Anwender, der auf den Server zugreift. Der Benutzer(Admin) verwaltet den Server. Der Benutzer kann, muss nicht Spieler sein.

Broadcast:
Die identische Nachricht an alle Teilnehmenden Roboter.

Deadlock:
Beim Deadlock können sich die Roboter nicht mehr ihrer Aufgabe nachkommen, weil sie sich gegenseitig behindern. Ein Deadlock ist also ein zu vermeidender Spielzustand.

GUI:
Die GUI ist die grafische Benutzeroberfläche für den Server und den Spielerclient.

Kartendaten:
Mit den Kartendaten sind die Informationen über das Wegenetz gemeint, welches als Graph implementiert wird.

KI:
Das Programm auf den Robotern, welches das Handeln des Roboters festlegt.

Konflikt:
Ein Konflikt tritt auf, wenn eine Straße von einem Roboter versperrt ist. Andere Roboter können diese zur selben Zeit nicht befahren.

NXT:
NXT ist ein Roboterbausatz der Firma Lego.

Ressourcen:
Unter Ressourcen fallen Industrien, Waren und der Kraftstoff für die Roboter.

Roboter:
Die Roboter arbeiten jeweils mit einer NXT-2.0-Einheit der Firma Lego und stellen die Transportmittel für die Waren dar.

Server:
Der Server dient hier als Koordinationsschnittstelle und übernimmt somit die Aufgabe des Spielleiters.

Spieler:
Menschlicher Anwender, der auf die Spieleroberfläche (nicht den Server) zugreift.

Spielzeit:
Die Spielzeit bezeichnet den Zeitraum, der nach Initialisierung des Spiels und dessen Start bis zum Erreichen des Spielziels abläuft.