ROSI -
Ein Simulationspaket für mobile Roboter

Handbuch

zur

Diplomarbeit

im Studiengang Softwaretechnik
am Fachbereich Elektrotechnik und Informatik
der Fachhochschule Hamburg

Vorgelegt von

Björn Gottfried

Referent: Prof. Dr. Owsnicki-Klewe

Korreferent: Prof. Dr. von Luck

Dezember 1996

Inhaltsverzeichnis

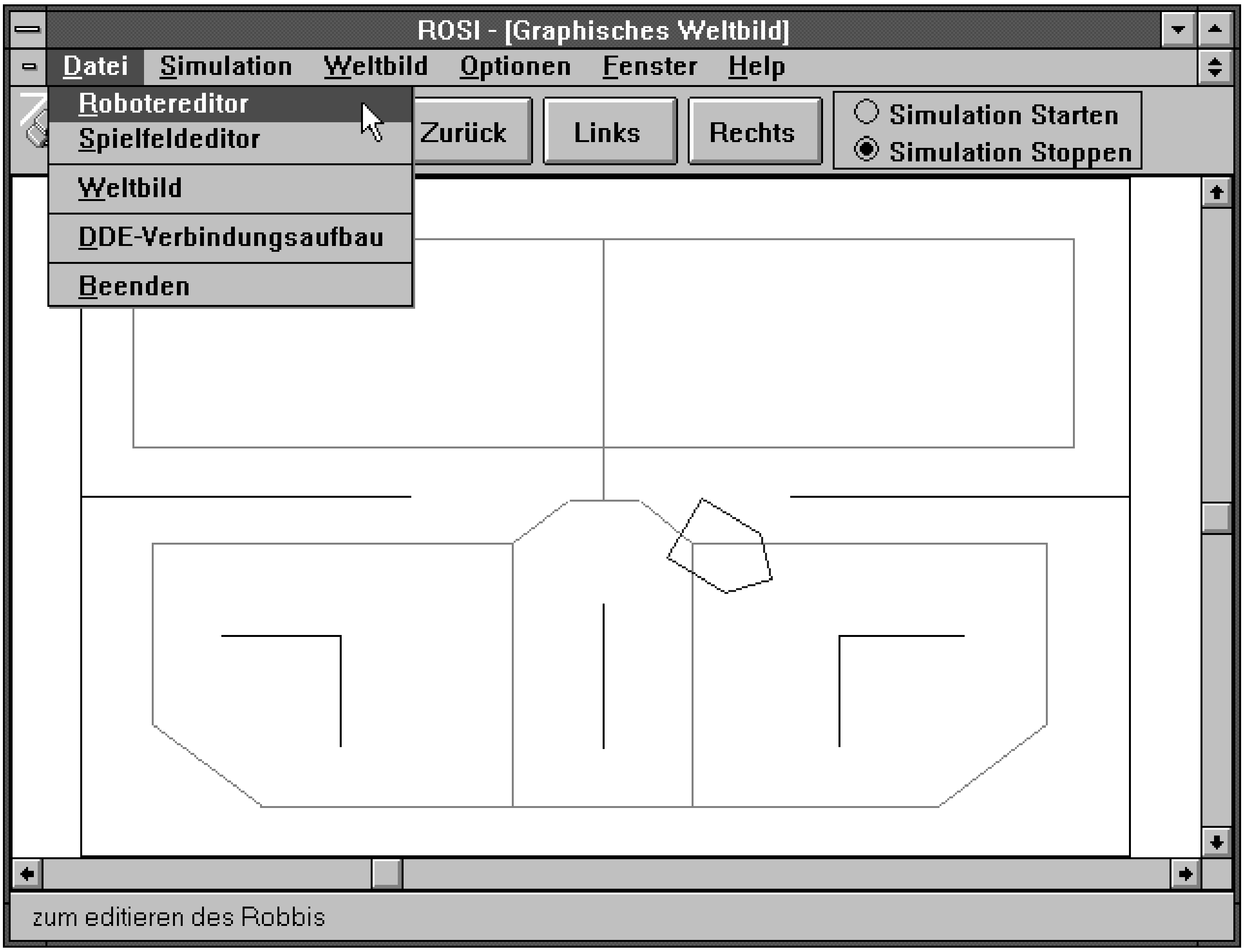
B. Handbuch

In diesem Handbuch werden die wichtigsten Funktionen von ROSI erklärt. Es wird gezeigt, wie Roboter und Spielfelder konfiguriert und wie Steuerungsprogramme mit dem Simulator getestet werden. Insbesondere sei an dieser Stelle auf den Menüpunkt Help im Programm hingewiesen. Denn auch im Hilfesystem werden die meisten Funktionen online erklärt.

Grundsätzlich ist zur Bedienung zu sagen, daß ROSI als **M**ultiple-**D**ocument-**I**nterface entwickelt worden ist: diese Benutzerschnittstelle beinhaltet außer einem Pulldownmenü eine Toolbar, für einige Funktionalitäten, und eine Statuspane, zur Anzeige einiger nützlicher Informationen.
Das Programm enthält eine Arbeitsfläche, auf der beliebig viele Fenster geöffnet, geschlossen und in ihrer Größe verändert werden können. Bei ROSI handelt es sich bei diesen Fenstern um Weltbilder, Robotereditoren und Spielfeldeditoren. Weltbilder visualisieren die Simulation einschließlich Roboter und Spielfeld, Robotereditoren sind zur Konfiguration der Roboter und Spielfeldeditoren sind zur Konfiguration der Spielfelder. Pro Fenstertyp existiert ein kontext-abhängiges Menü.
Außerdem existiert ein Windows Hilfesystem. ROSI läßt sich also wie die meisten Windowsprogramme bedienen.

B.1 Programmstart

Um den Simulator zu starten ist die Datei ROSI.EXE unter Windows 3.1 oder Windows 95 aufzurufen. Der Aufruf kann einige Sekunden dauern.

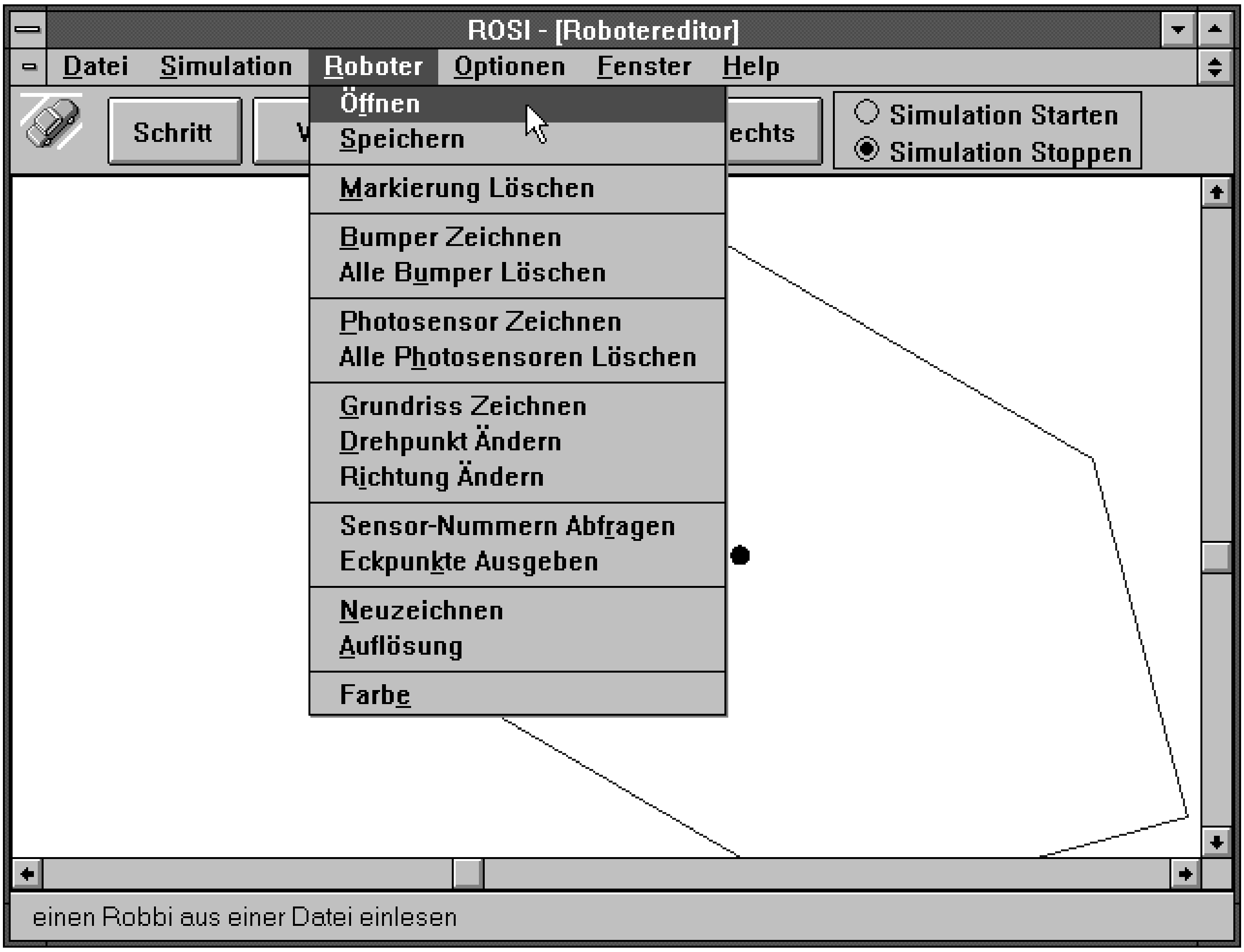


B.2 Robotereditor

Der Robotereditor wird mit dem Menüpunkt Datei / Robotereditor geöffnet. Mit dem Robotereditor werden alle möglichen Roboterkonfigurationen eingestellt.

Es können parallel mehrere Robotereditoren geöffnet werden, um z.B. beim editieren verschiedene Roboterkonfigurationen vergleichen zu können. Solange aber noch wenigstens ein Robotereditor offen ist, kann die Simulation nicht gestartet werden.

Der Robotereditor, der zuletzt geschlossen wird, ist für die Simulation maßgebend. D.h. der Roboter, der in diesem Robotereditor erstellt wurde, wird simuliert. Dazu erhält jedes geöffnete Weltbild diesen Roboter.

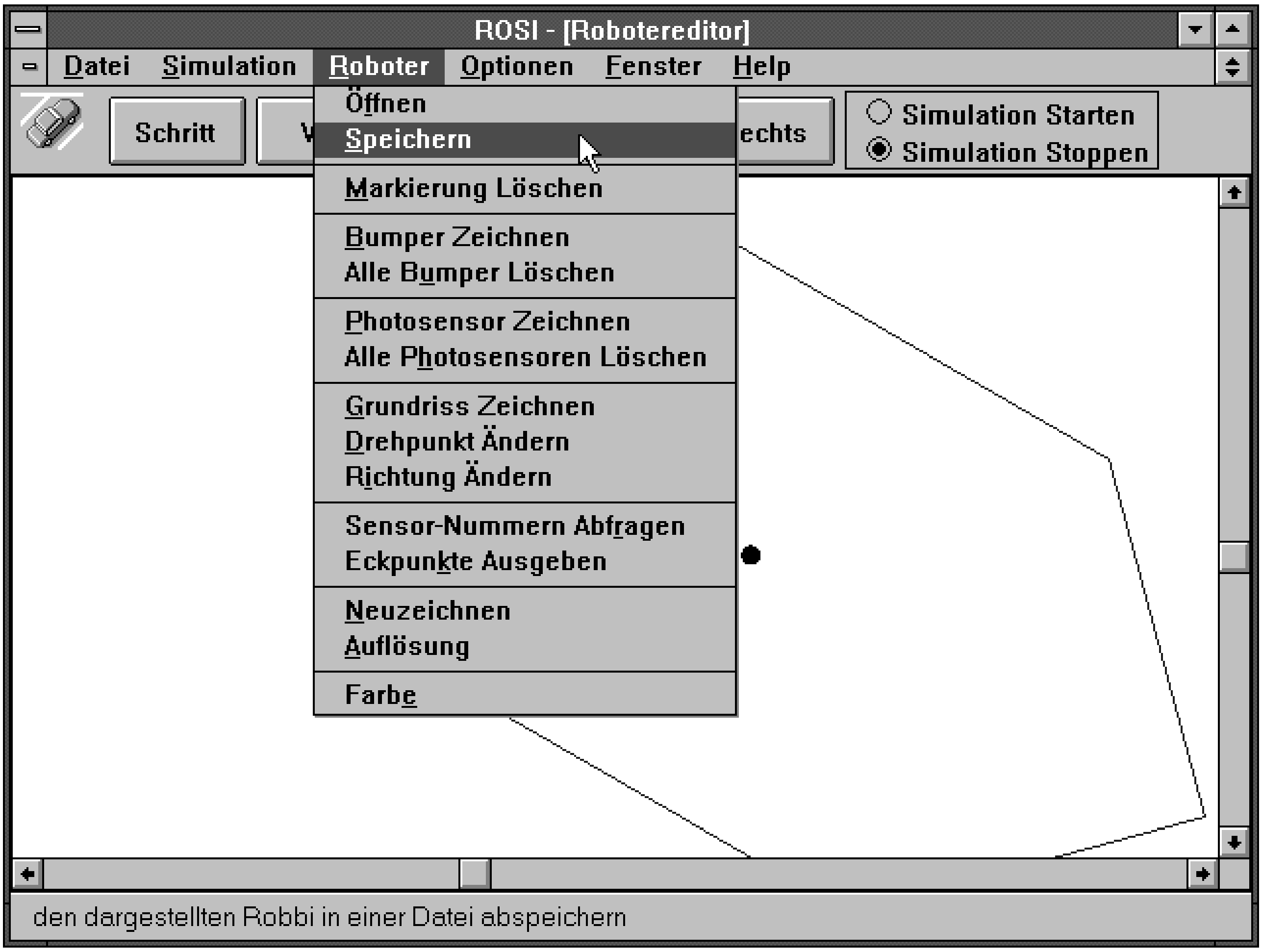


B.2.1 Roboterdatei öffnen

Ein Roboter, der zuvor mit dem Menüpunkt Roboter / Speichern gespeichert wurde, kann mittels Roboter / Öffnen in den Robotereditor geladen werden.

Der Roboter, der bisher im Editor zu sehen war, wird gelöscht.

Aus der Datei werden Grundriß, Drehpunkt, Ausrichtung, Mikroschalter und Photosensoren des Roboters gelesen.

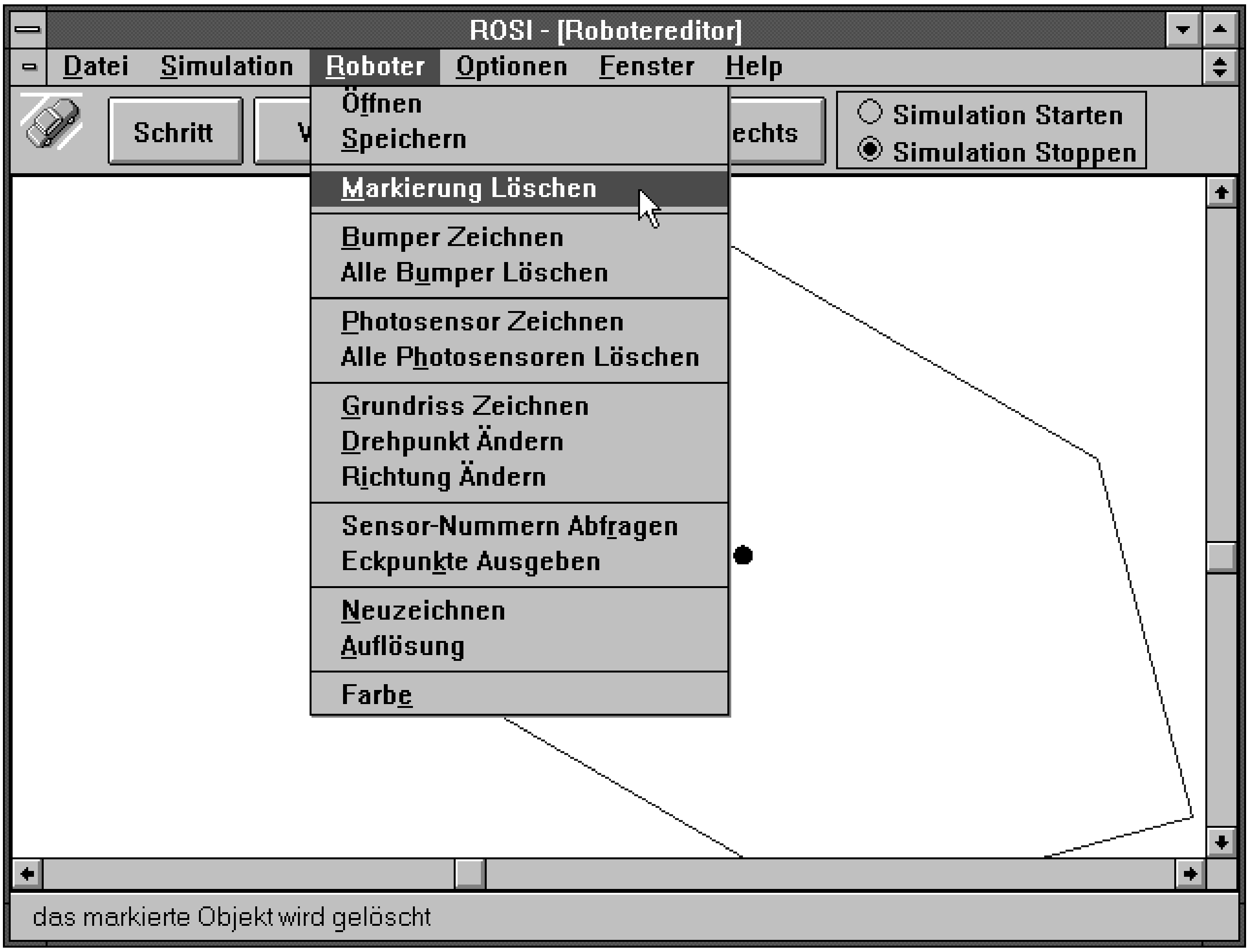


B.2.2 Roboter speichern

Mittels Roboter / Speichern kann der Roboter, der gerade im Editor zu sehen ist, gespeichert werden.

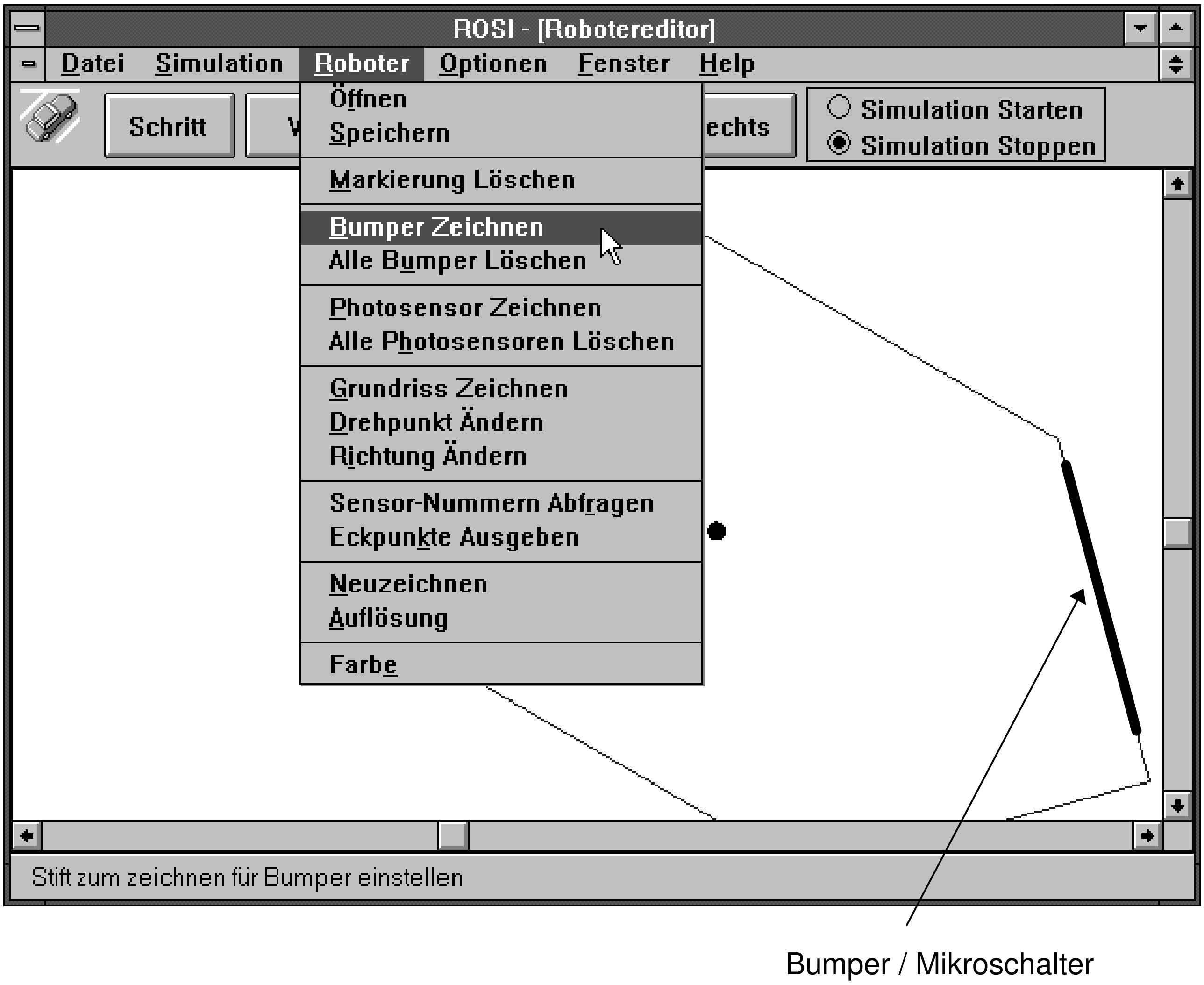
Der Roboter wird in einer Textdatei gespeichert. D.h. es werden die Koordinaten seines Grundrisses, die Koordinaten seines Drehpunktes, seine Ausrichtung und die Koordinaten seiner Sensoren gespeichert. Die Textdateien der Roboter können in Texteditoren editiert werden, um z.B. auf beliebig viele Nachkommastellen genau die Koordinaten aller Eigenschaften des Roboters anzugeben.

Mittels Roboter / Öffnen kann ein auf diese Weise gespeicherter Roboter wieder in den Robotereditor geladen werden.



B.2.3 Markierungen löschen

Mikroschalter und Photosensoren kann man markieren, indem man bei gedrückter Shift-Taste mit der linken Maustaste auf den Mikroschalter bzw. Photosensor drückt. Ein markiertes Element erscheint rot und kann mit dem Menüpunkt Roboter / Markierung Löschen gelöscht werden.

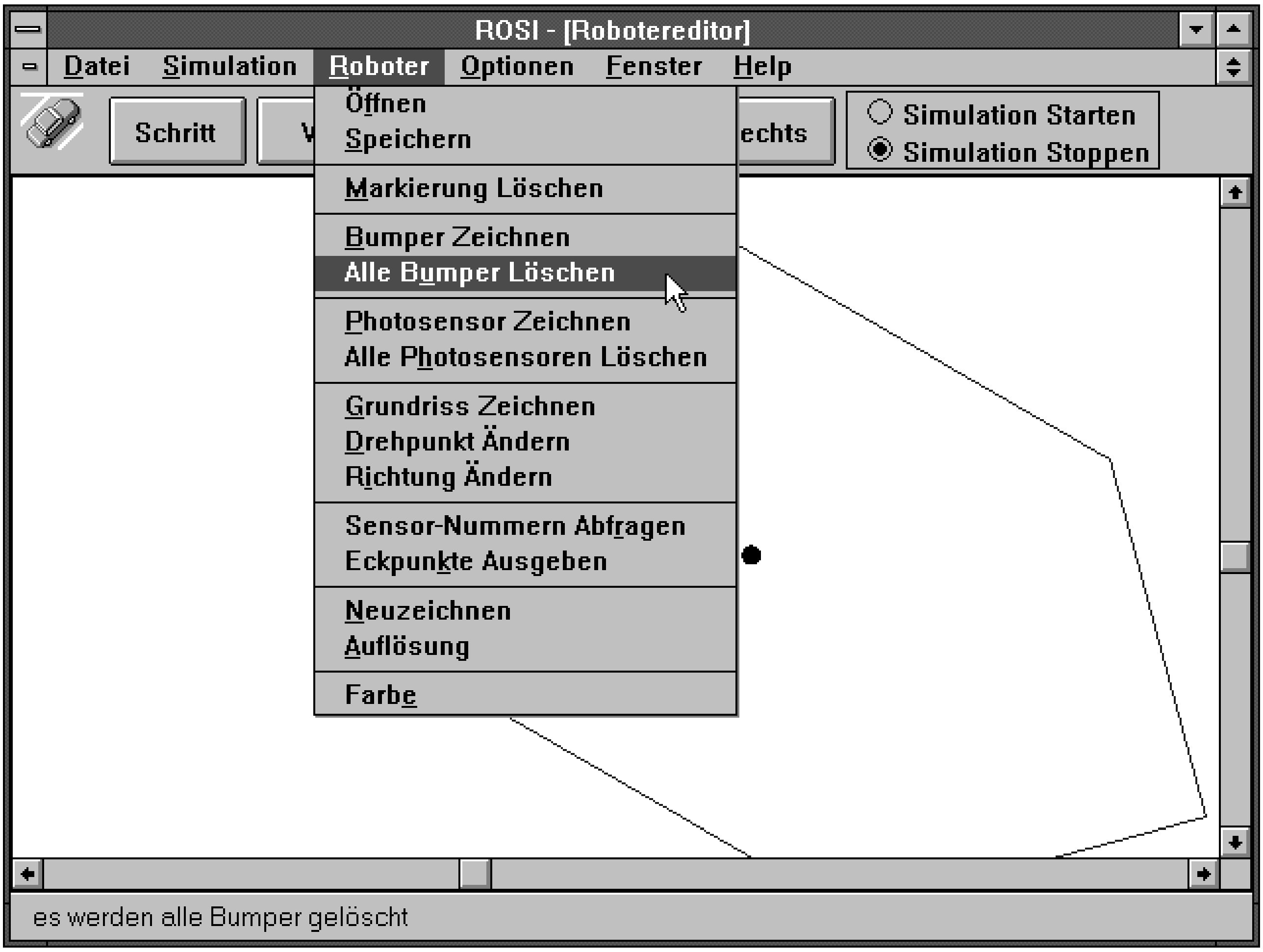


B.2.4 Mikroschalter zeichnen

Mittels Roboter / Bumper Zeichnen wird der Mauscursor in die Lage versetzt, einen Mikroschalter zu zeichnen. Die Begriffe *Bumper* und *Mikroschalter* werden in diesem Handbuch und im Programm synonym verwendet.

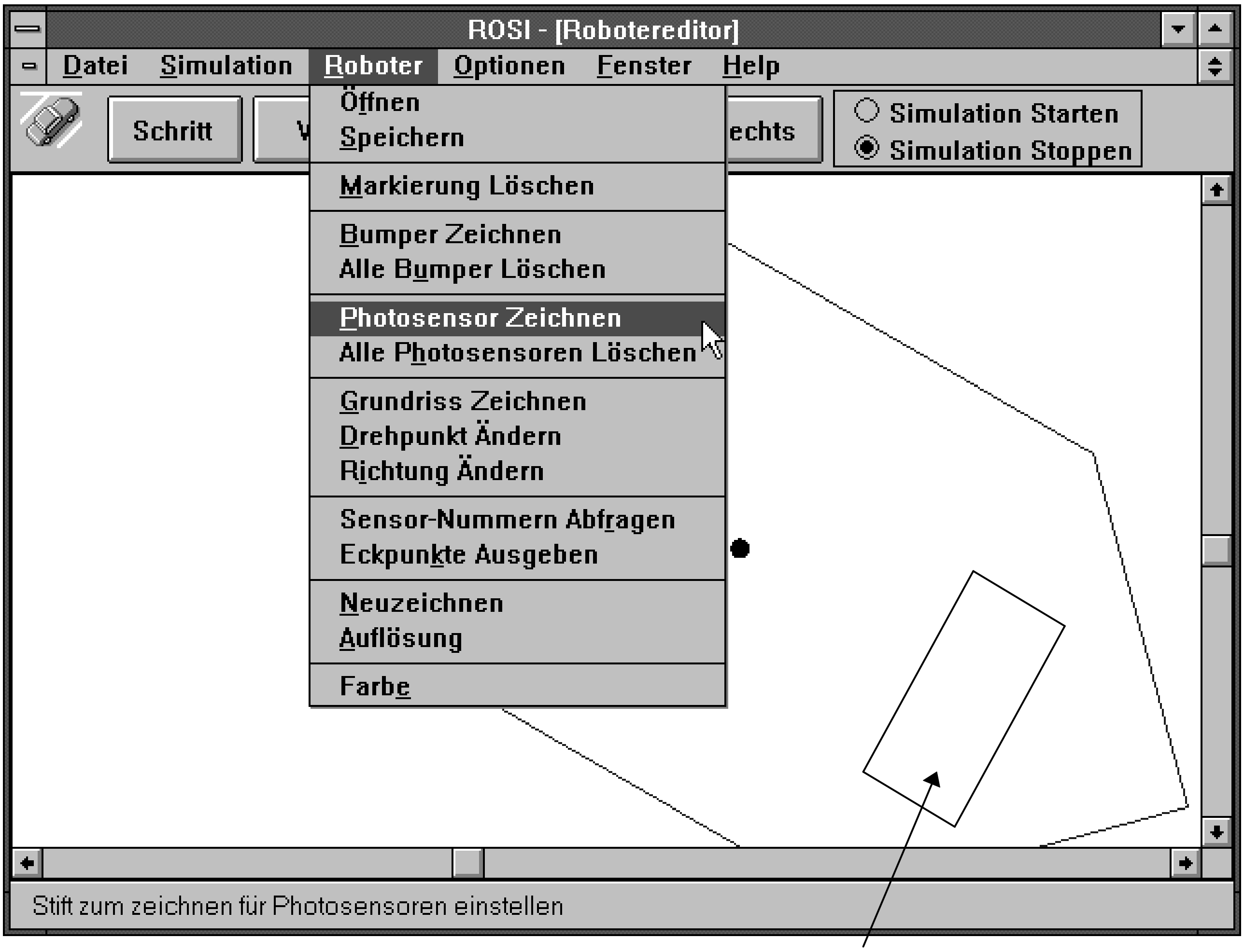
Der Mauscursor ist nun in der Lage eine dicke türkise Linie zu zeichnen. Dazu drückt man die linke Maustaste und bewegt die Maus so, daß man die Linie in gewünschter Art zeichnet; man läßt die Maustaste los, wenn die Seite fertiggezeichnet ist.
Derartige Linien kann man allerdings nur auf dem Grundriß zeichnen. Alle Linien, die man woanders zu zeichnen versucht, werden gelöscht. Man braucht aber die Linien des Grundrisses nicht genau zu treffen. Ein kleiner Toleranzspielraum wird berücksichtigt, um das Zeichnen der Mikroschalter auf dem Grundriß zu erleichtern.
Solange kein anderer Menüpunkt zwischendurch ausgewählt wurde, kann man hintereinander mehrere Mikroschalter zeichnen.

Ein fertiggezeichneter Mikroschalter wird als dunkelblaue dicke Linie dargestellt.



B.2.5 Alle Mikroschalter löschen

Mit Hilfe des Menüpunktes Roboter / Alle Bumper löschen werden mit einem Schlag alle Bumper gelöscht. Unter Umständen hat man bereits viele Sensoren gezeichnet, die man aber alle wieder verwerfen möchte. Das einzelne Löschen aller Sensoren wäre zu aufwendig, da man jeden Sensor einzeln markieren und mit dem Menüpunkt Roboter / Markierung Löschen löschen müßte.



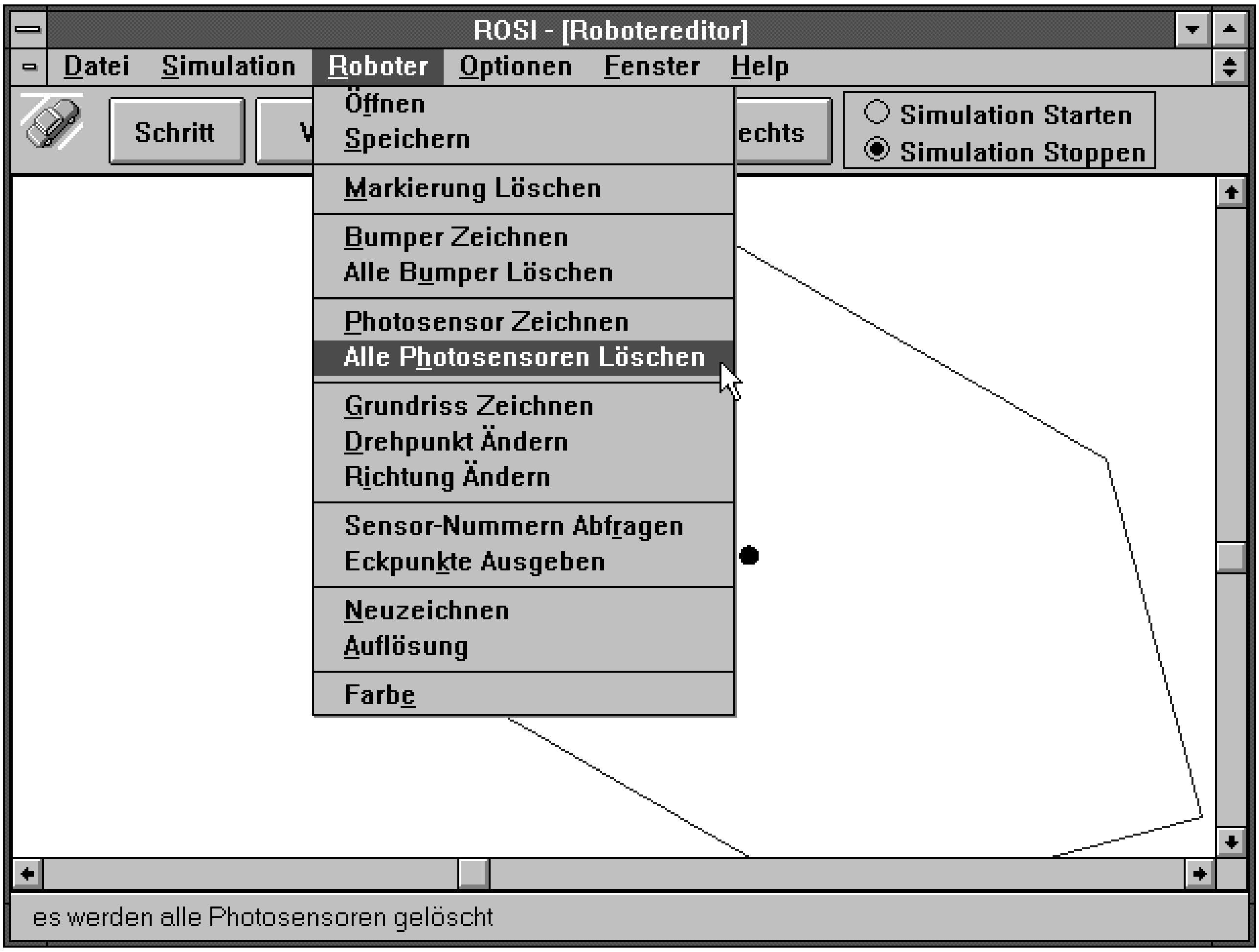
Photosensor

B.2.6 Photosensor zeichnen

Mittels Roboter / Photosensor Zeichnen wird der Mauscursor in die Lage versetzt, einen Photosensor zu zeichnen.

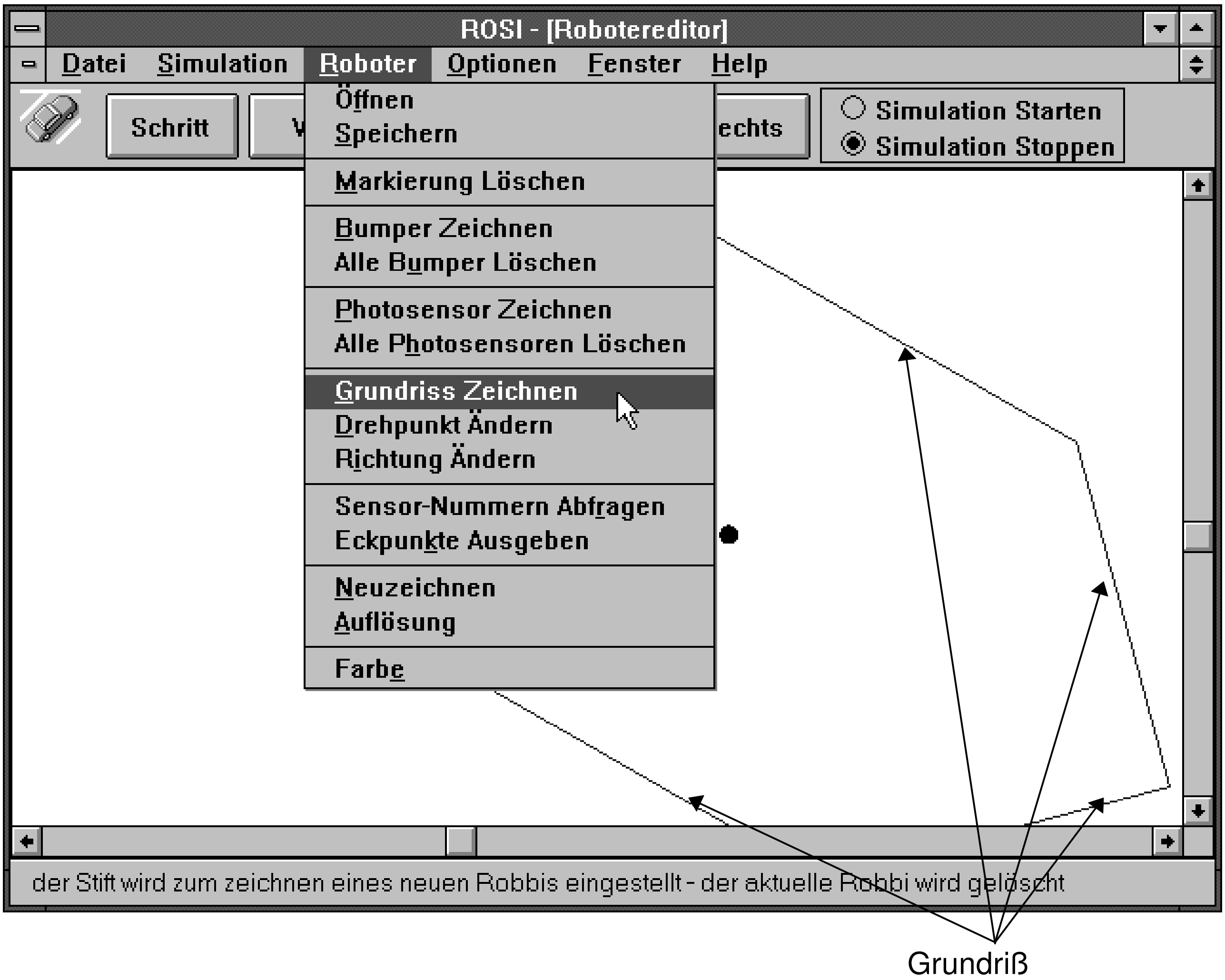
Man drückt die linke Maustaste und bewegt die Maus so, daß man eine türkise Polygonseite zeichnet; man läßt die Maustaste los, wenn die Seite fertiggezeichnet ist. Man kann die Maus nun zu beliebigen weiteren Punkten bewegen, die allesamt Eckpunkte des zu zeichnenden Polygons darstellen. An jedem Eckpunkt muß jeweils die linke Maustaste einmal gedrückt bzw. nur losgelassen werden, falls die Maustaste schon gedrückt war. Das Polygon wird abgeschlossen indem man zur linken Maustaste zusätzlich eine Shift-Taste drückt.

Ein fertiggezeichneter Photosensor wird als dünnes dunkelrosa Polygon dargestellt.



B.2.7 Alle Photosensoren löschen

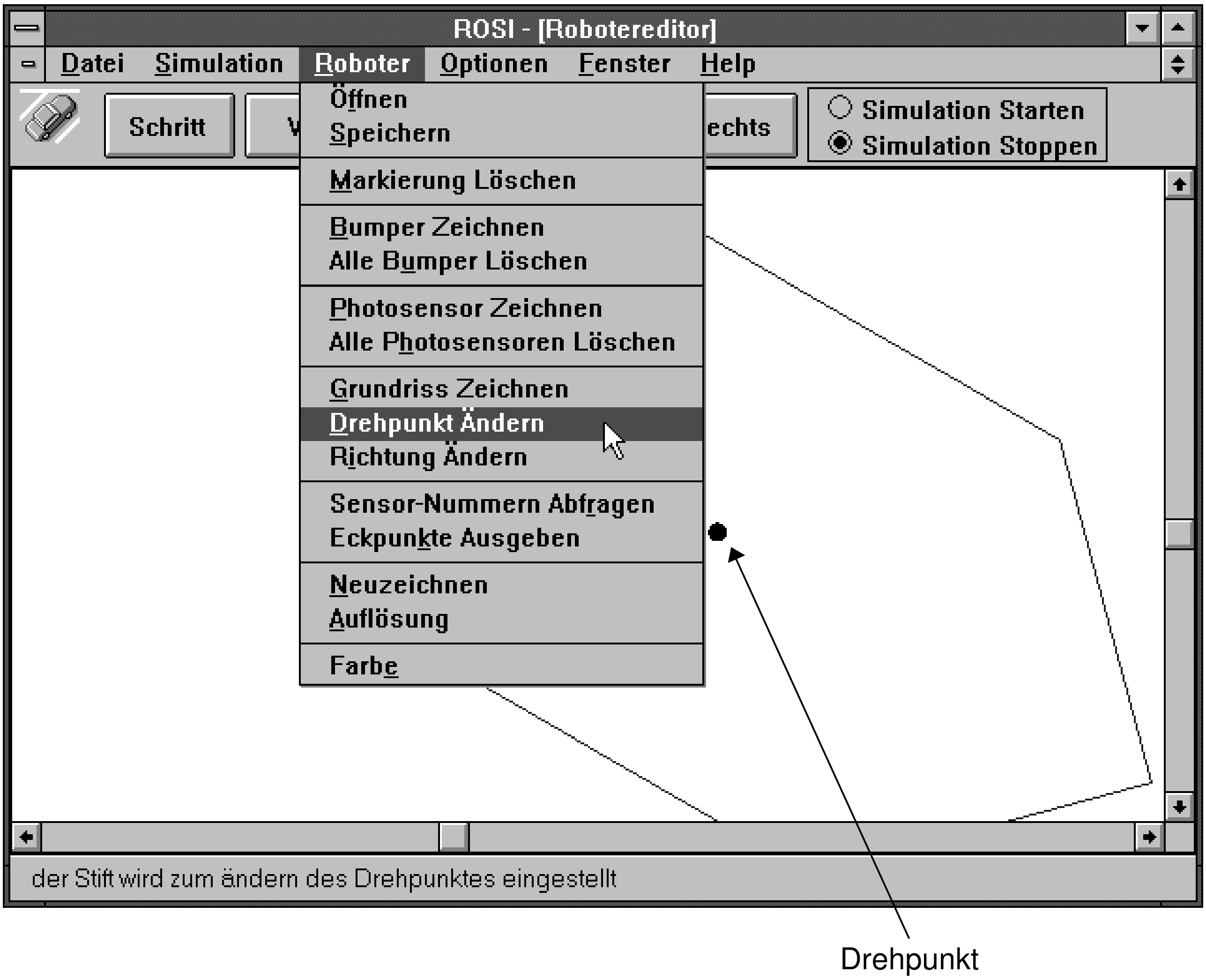
Mit Hilfe des Menüpunktes Roboter / Alle Photosensoren löschen werden mit einem Schlag alle Sensoren gelöscht. Unter Umständen hat man bereits viele Sensoren gezeichnet, die man aber alle wieder verwerfen möchte. Das einzelne Löschen aller Sensoren wäre zu aufwendig, da man jeden Sensor einzeln markieren und mit dem Menüpunkt Roboter / Markierung Löschen löschen müßte.



B.2.8 Robotergrundriß zeichnen

Um den Grundriß des Roboters zu zeichnen, wählt man Roboter / Grundriss zeichnen aus. Das Fenster wird gelöscht und der Mauscursor ist in der Lage, ein Polygon zu zeichnen.
Dazu drückt man die linke Maustaste und bewegt die Maus so, daß man eine türkise Polygonseite zeichnet; man läßt die Maustaste los, wenn die Seite fertiggezeichnet ist. Man kann die Maus nun zu beliebigen weiteren Punkten bewegen, die allesamt Eckpunkte des zu zeichnenden Polygons darstellen. An jedem Eckpunkt muß jeweils die linke Maustaste einmal gedrückt bzw. nur losgelassen werden, falls die Maustaste schon gedrückt war. Das Polygon wird abgeschlossen indem man zur linken Maustaste zusätzlich eine Shift-Taste drückt.

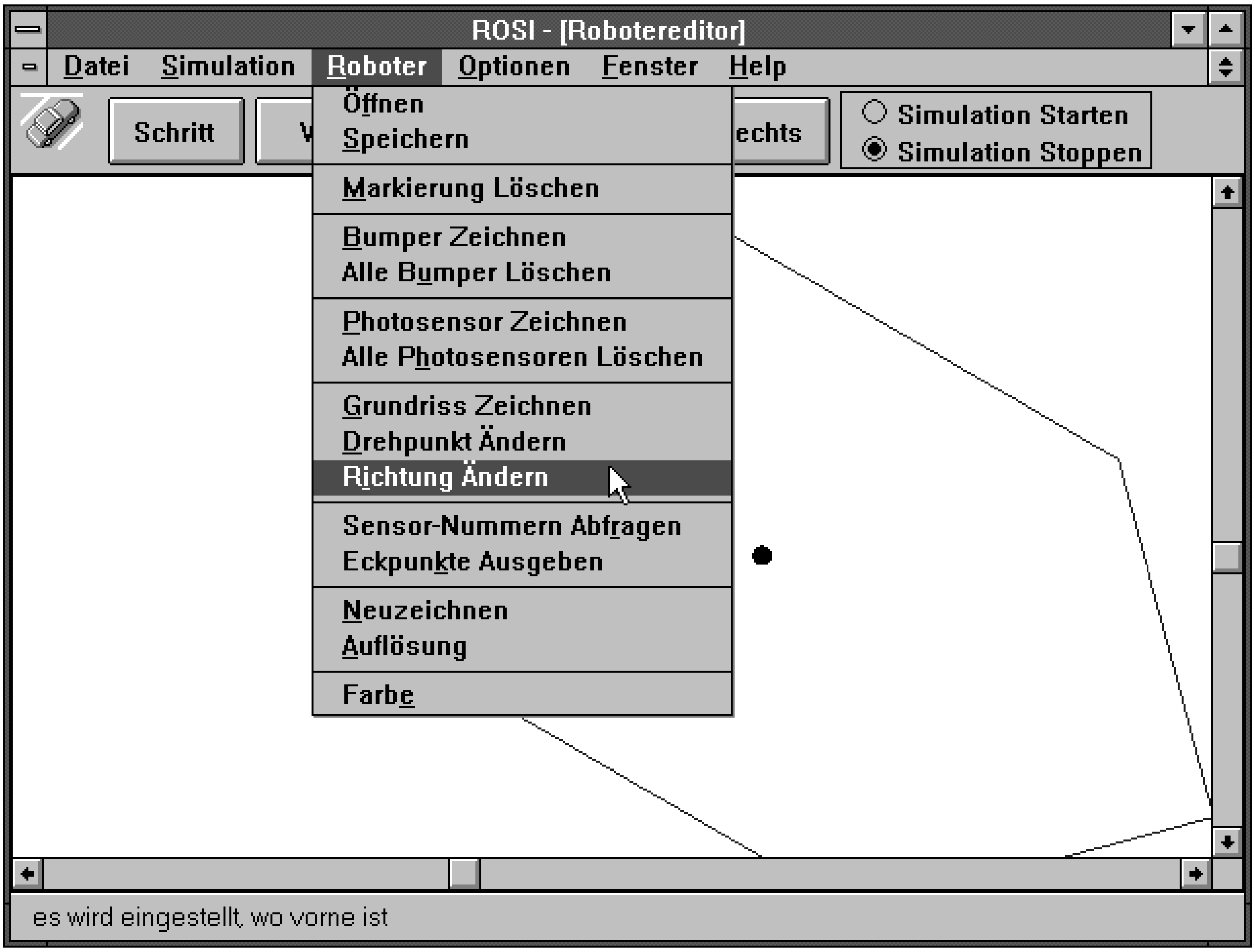
Ein fertiggezeichneter Grundriß wird als dunkelblaue dünne Linie dargestellt.



B.2.9 Roboterdrehpunkt ändern

Durch Roboter / Drehpunkt Ändern wird die Maus in den Zustand gesetzt, den Drehpunkt versetzen zu können. Der Drehpunkt ist der große schwarze Punkt im Robotereditor, der meistens im Polygon des Robotergrundrisses sitzt. Man drückt eine Shift-Taste und gleichzeitig die linke Maustaste auf diejenige Position, wo der neue Drehpunkt sitzen soll.

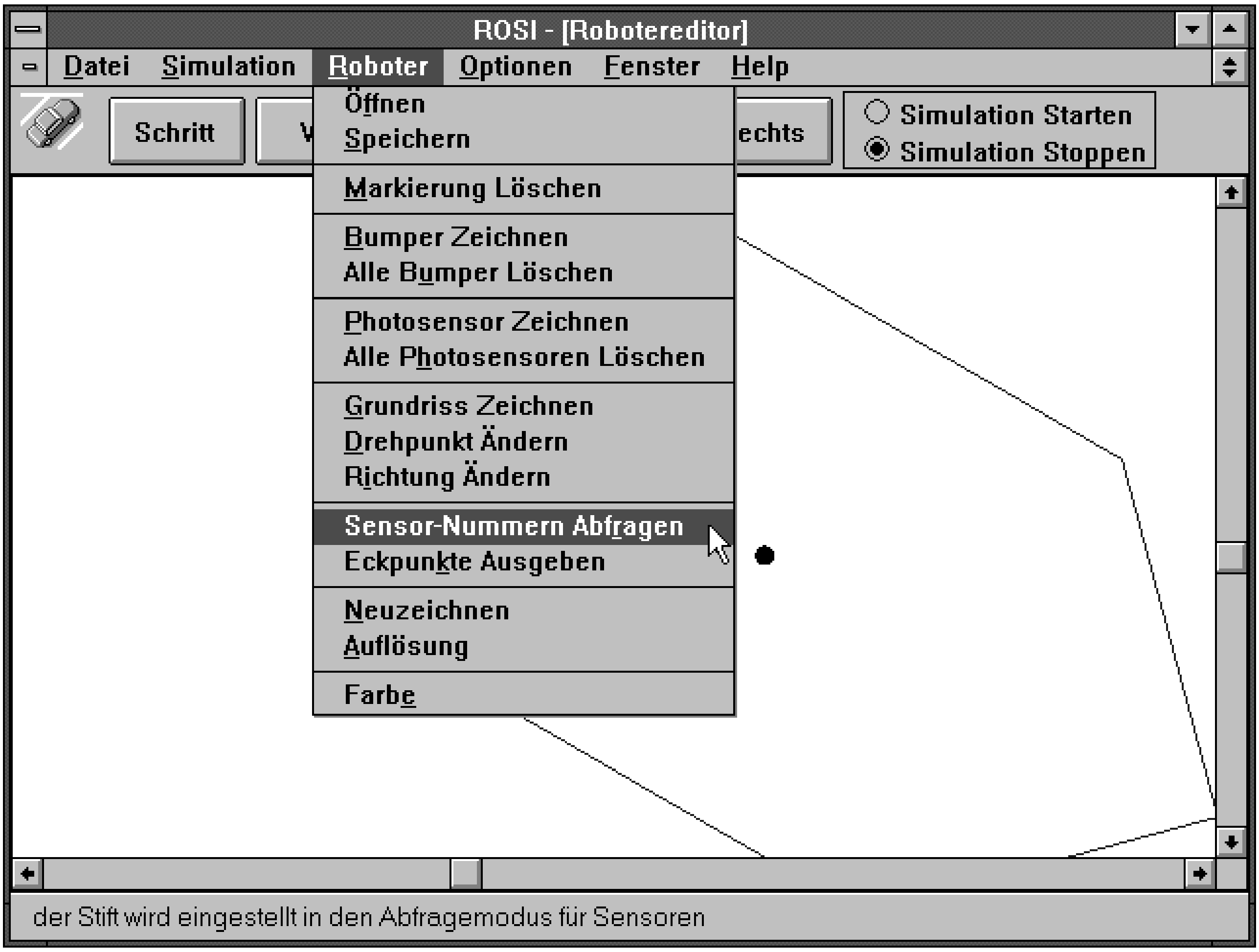
Der Drehpunkt stellt die Achse dar, um die sich der Roboter dreht. Der Drehpunkt kann überall sitzen.



B.2.10 Roboterrichtung ändern

Mittels Roboter / Richtung Ändern wird die Vorderseite des Roboters bestimmt. Oben steht für 0° und rechts für 90°. Entsprechend sind alle anderen Richtungen im Uhrzeigersinn bestimmt. Wenn beispielsweise 90° angegeben wird, heißt das, daß der Roboter auf dem Bildschirm nach rechts fährt, wenn er geradeaus fahren soll. Die rechte Seite des Grundrisses stellt in diesem Fall die Vorderseite des Roboters dar.

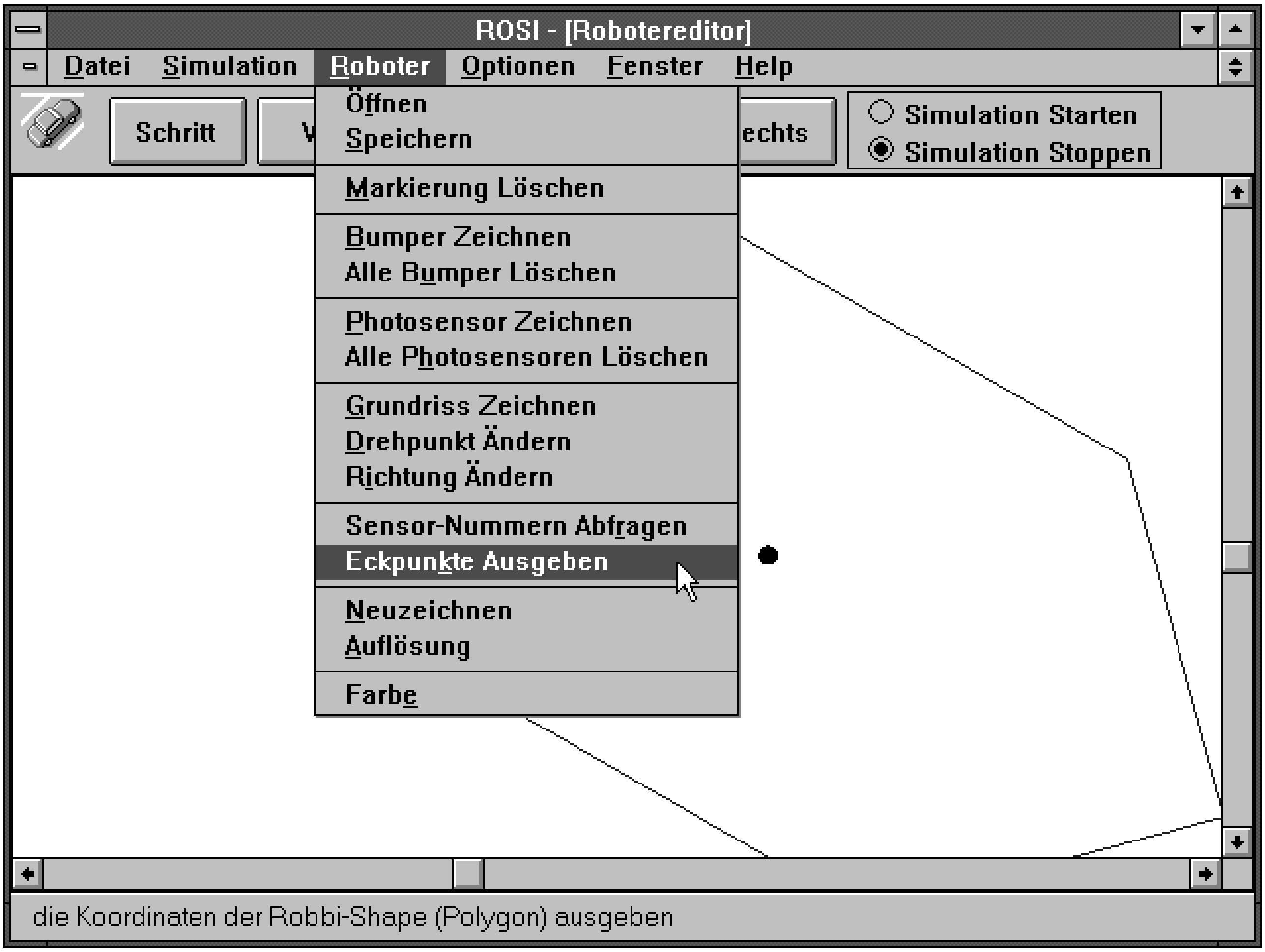
Man kann ausprobieren, wo die Vorderseite ist. Dazu startet man die Simulation und gibt über die Toolbar an, daß sich der Roboter Vor bewegen soll. Man sieht dann, wohin sich der Roboter bewegt. Die Bewegungsrichtung zeigt seine Vorderseite an.



B.2.11 Sensornummern ermitteln

Mit dem Menüpunkt Roboter / Sensor-Nummern Abfragen können die Snsornummern ermittelt werden. Dazu doppelklickt man auf den entsprechenden Sensor. Es geht ein Informationsfenster auf, das anzeigt, um was für einen Sensor es sich handelt und was für eine Nummer dieser Sensor hat.
Solange kein anderer Menüpunkt ausgewählt wurde, können die Sensornummern abgefragt werden.

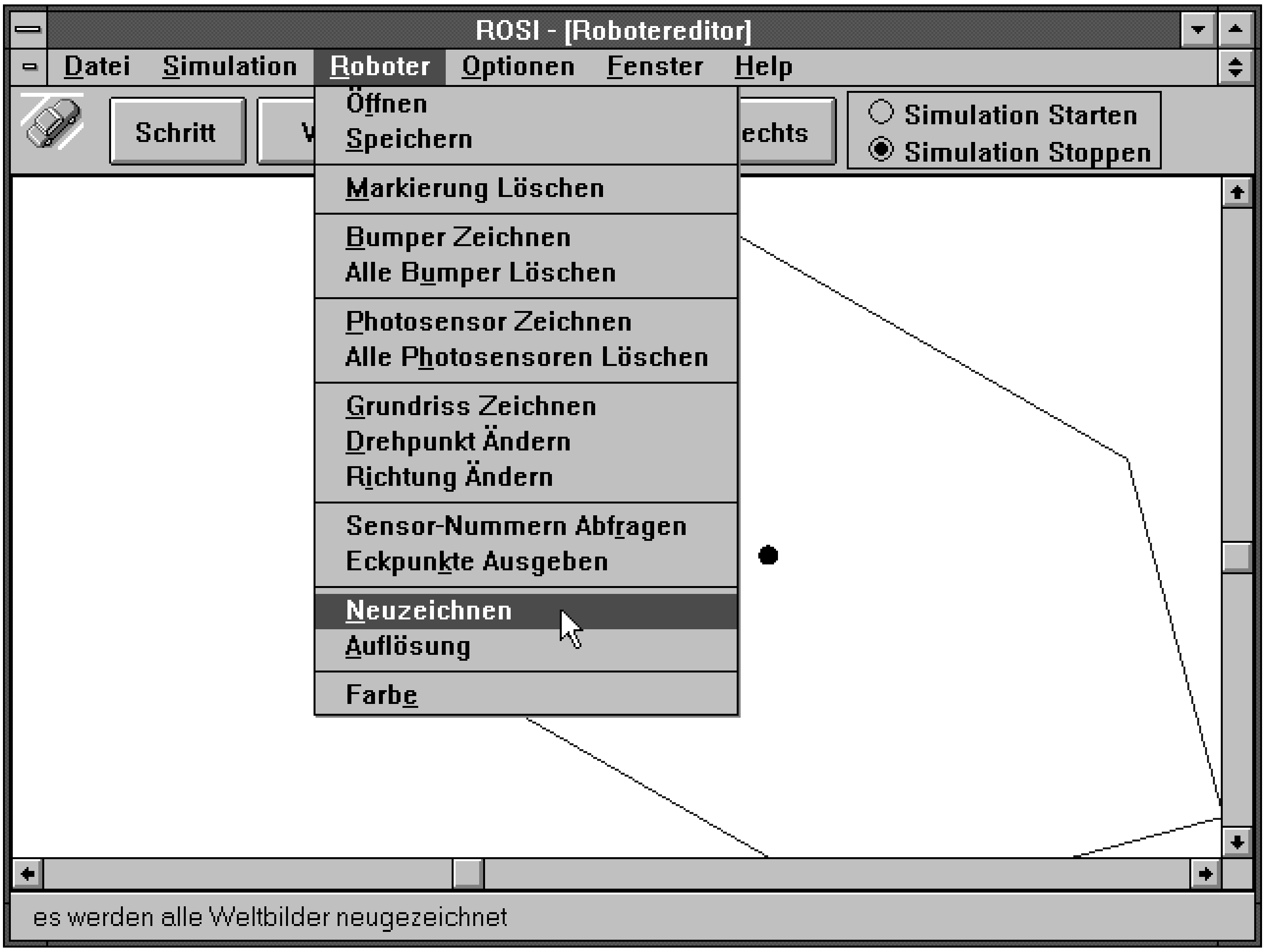
Die Sensornummern werden automatisch vergeben und können nicht vom Benutzer selbst bestimmt werden. Die Vergabe der Nummern erfolgt bei der Erstellung eines Sensors. Es wird jeweils die kleinste noch freie Nummer vergeben. Die Nummern beginnen bei 1.



B.2.12 Koordinaten des Robotergrundrisses

Um sich die Koordinaten des Robotergrundrisses anzugucken wählt man diesen Menüpunkt aus. Es werden in Informationsfenstern nacheinander alle reellen Koordinaten des Grundrisses angezeigt.

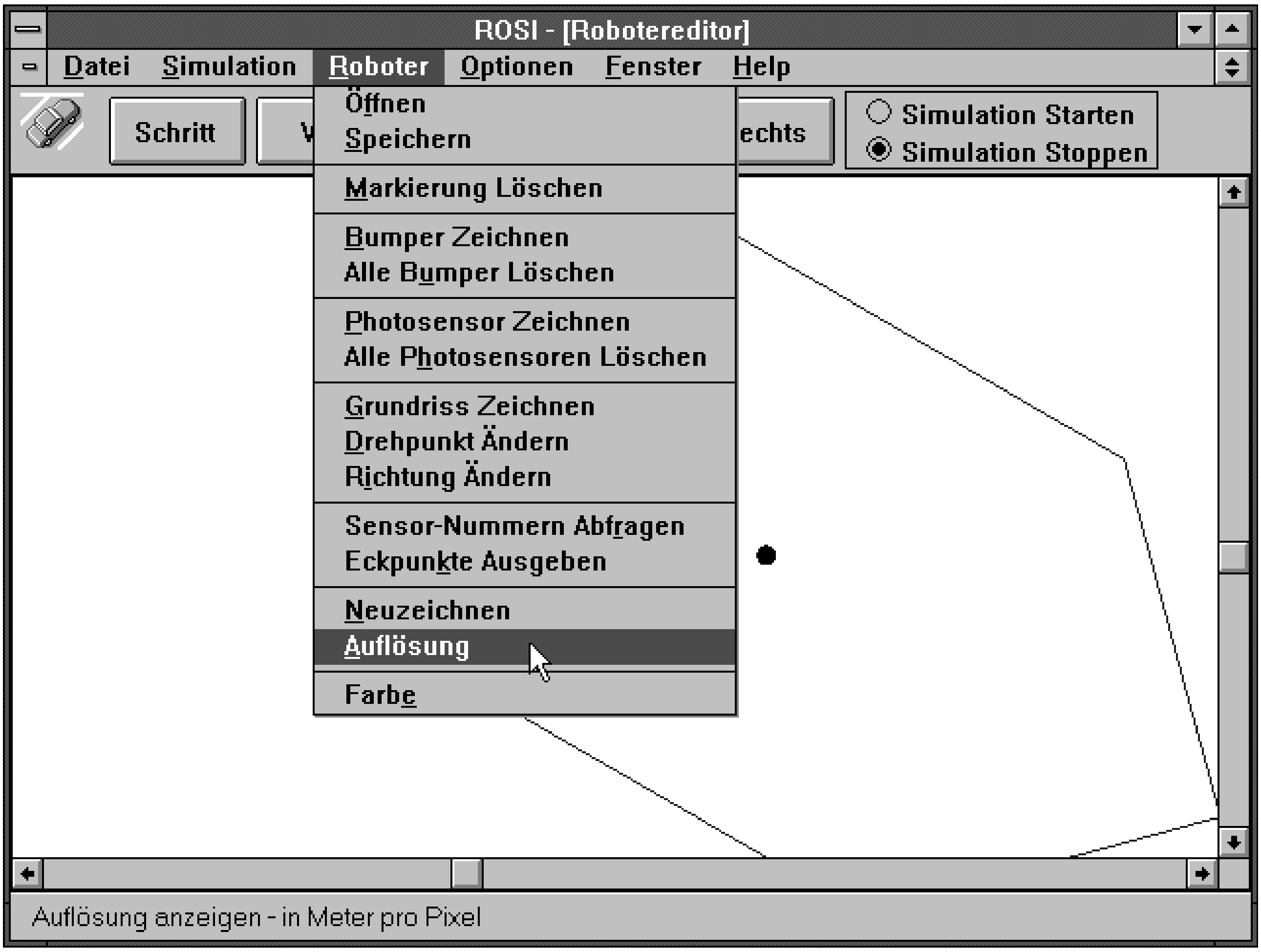
Da die auf dem Bildschirm gezeichneten Polygone immer etwas ungenau dargestellt werden, kann man auf diese Weise die genauen Koordinaten des Roboters erfragen. Unter Umständen genügt einem die Genauigkeit nicht und man möchte den Roboter in einer Datei speichern, um die Koordinaten in einem Texteditor genauer zu editieren.



B.2.13 Neuzeichnen

Mit diesem Menüpunkt wird lediglich alles einmal komplett neu gezeichnet.

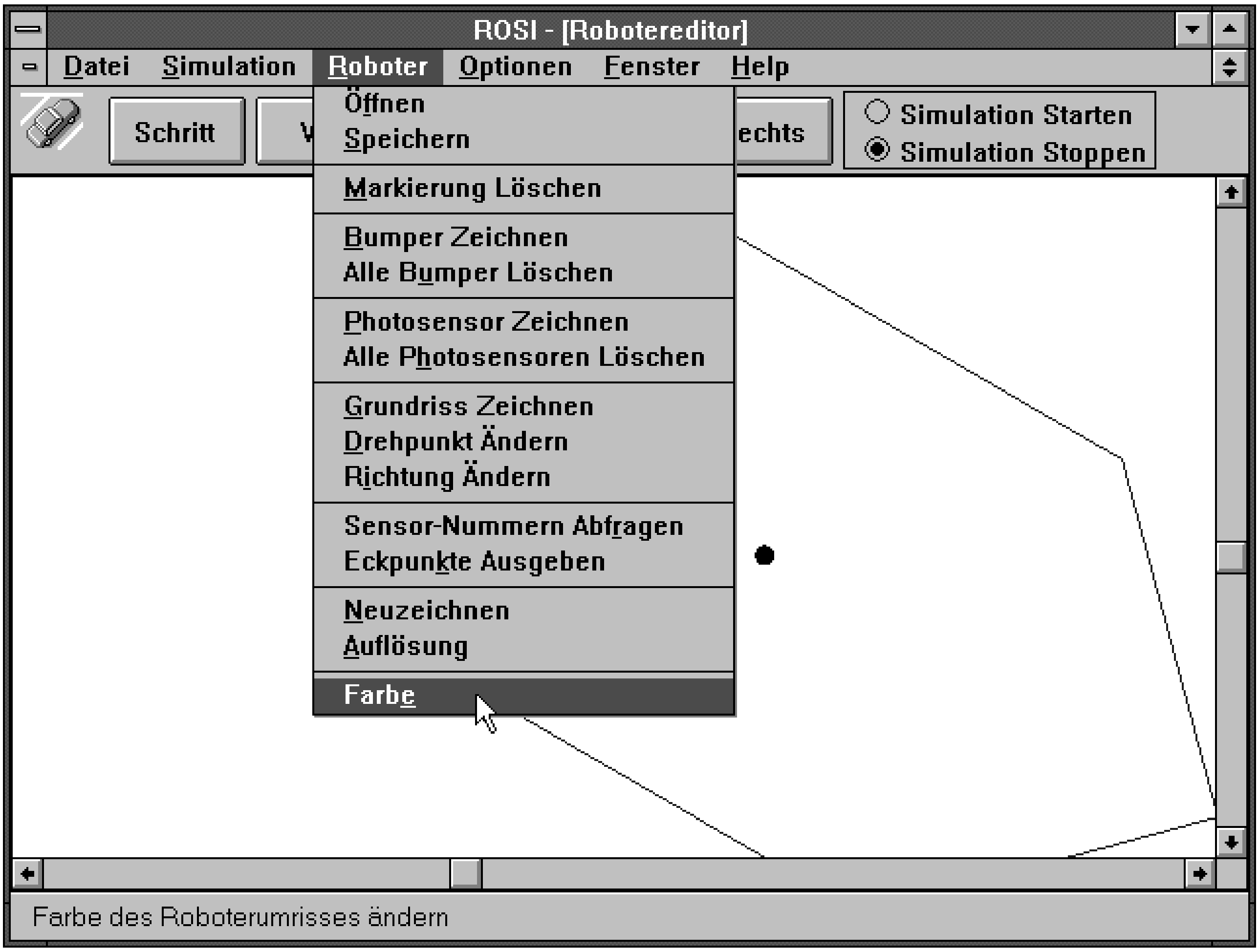
Das ist manchmal sinnvoll, falls irgendwelche graphischen Darstellungsprobleme, wie beispielsweise nicht korrekt wiederhergestellte Fenster, auftreten. Hierdurch werden keine Änderungen am Roboter vorgenommen.



B.2.14 Auflösung anzeigen

Mit diesem Menüpunkt wird die Darstellungsgenauigkeit angezeigt. Die Auflösung wird in Metern pro Pixel angegeben.

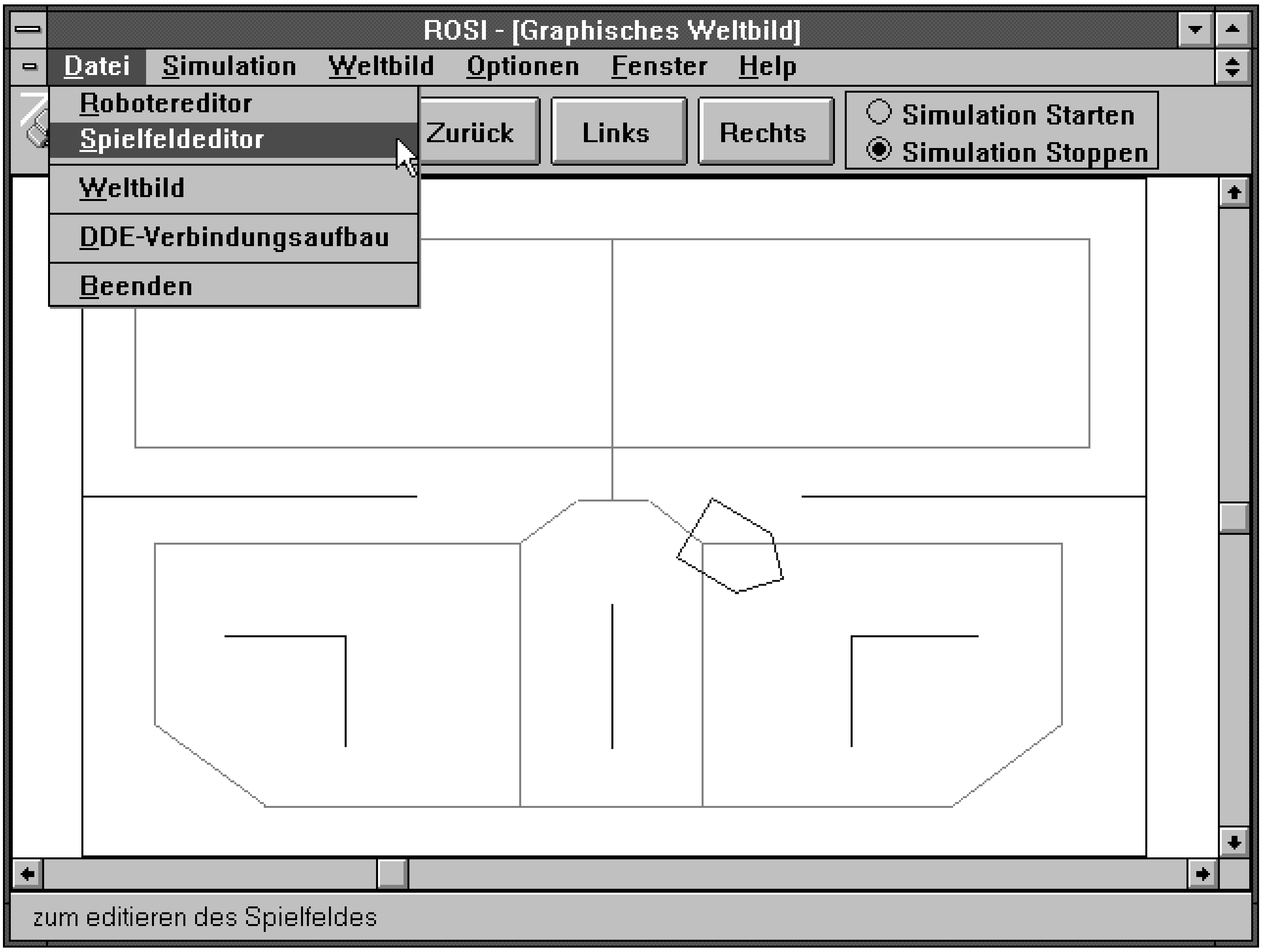
0,00075 Meter/Pixel bedeutet, daß auf ein Pixel ungefähr 0,00075 Meter kommen.



B.2.15 Farbe des Roboters festlegen

Mit diesem Menüpunkt wird die Farbe des Roboters festgelegt. Genauer gesagt betrifft das nur die Farbe des Robotergrundrisses, da nur der Grundriß dargestellt wird.

Es wird dazu der Windows gebräuliche Farbdialog geöffnet.

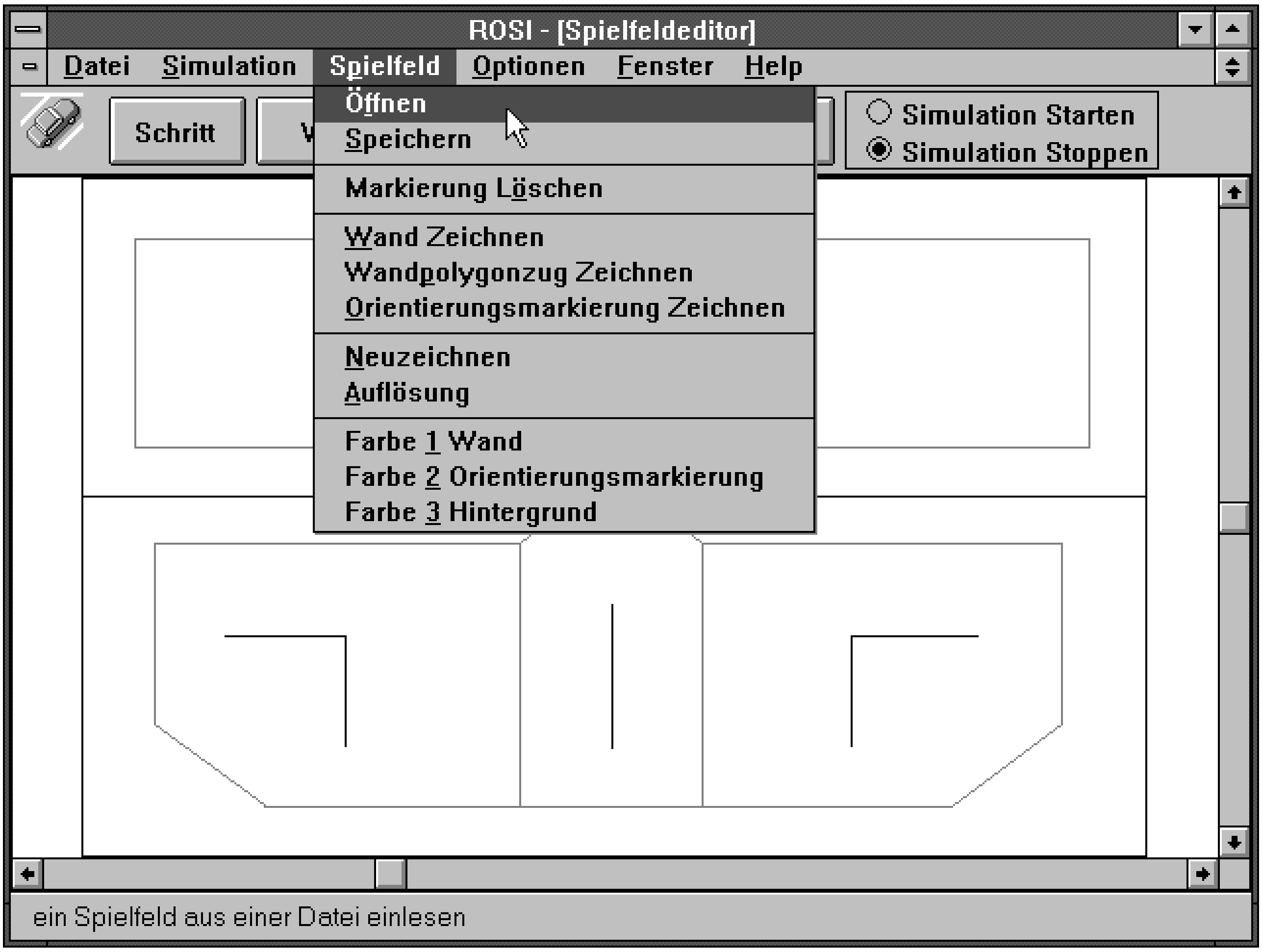


B.3 Spielfeldeditor

Den Spielfeldeditor startet man mit dem Menüpunkt Datei / Spielfeldeditor. Er dient dazu alle möglichen Spielfeldkonfigurationen einzustellen. D.h. es können Wände und Orientierungsmarkierungen gezeichnet, gelöscht und alle möglichen Farben geändert werden.

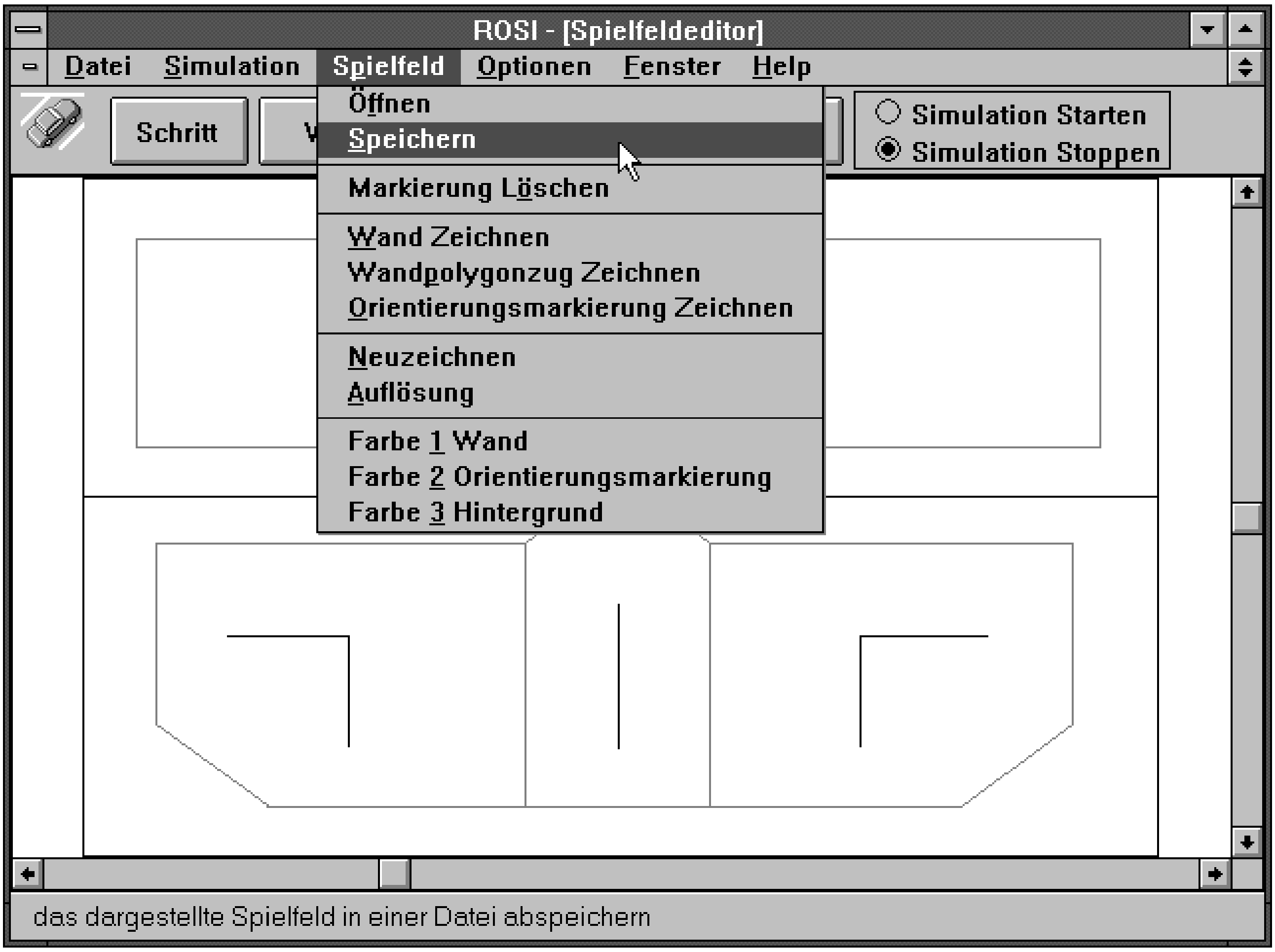
Es können parallel mehrere Spielfeldeditoren geöffnet sein, um unterschiedliche Spielfeldkonfigurationen vergleichen zu können.

Sobald der Spielfeldeditor geschlossen wird, wird das entsprechende Spielfeld, das zuletzt im Spielfeldeditor zu sehen war, in allen Weltbildern übernommen.



B.3.1 Spielfeld laden

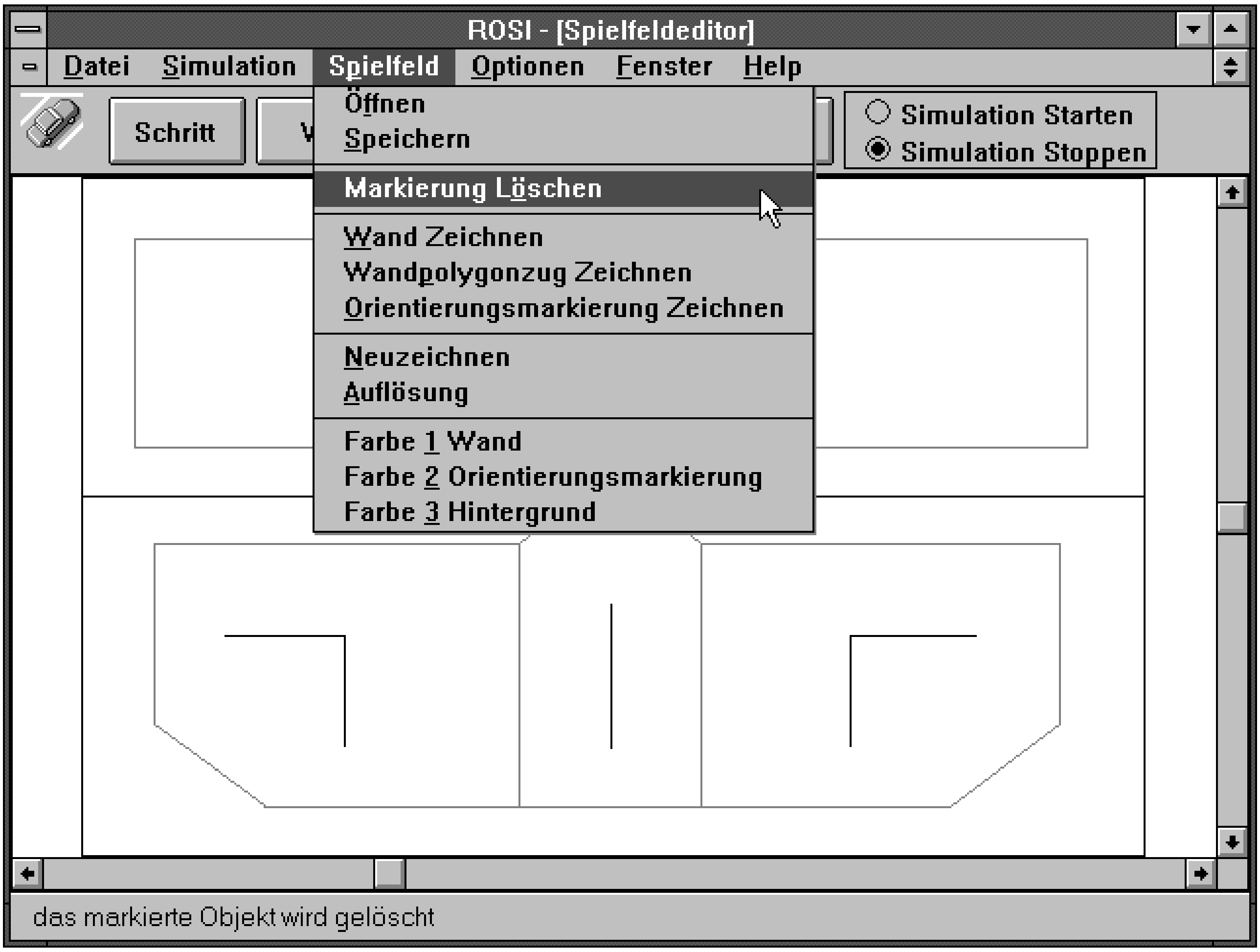
Mittels Spielfeld / Öffnen können alle Spielfeldeigenschaften aus einer Datei geladen werden, die zuvor mittels Spielfeld / Speichern erstellt wurde.



B.3.2 Spielfeld speichern

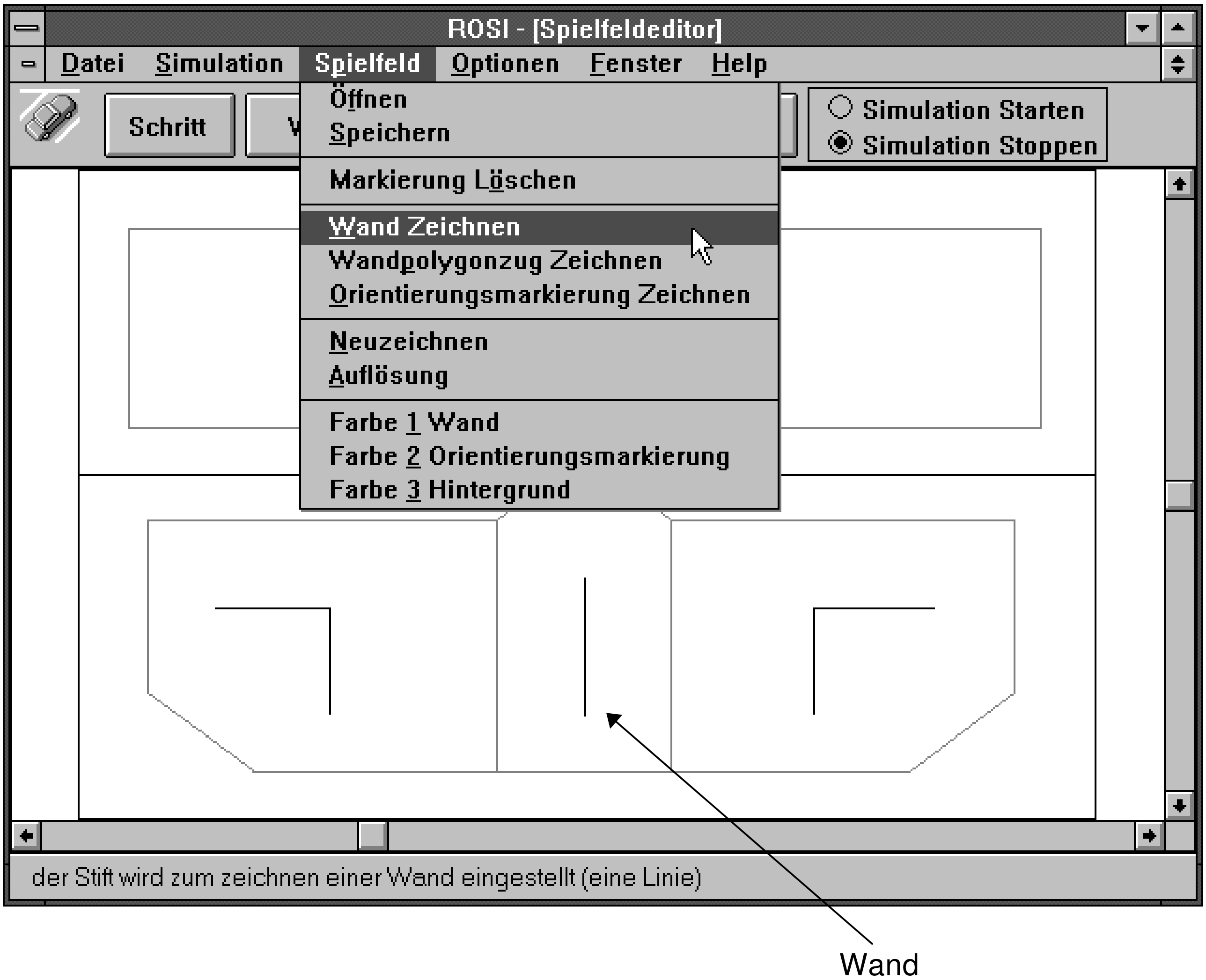
Mittels Spielfeld / Speichern kann das Spielfeld, das gerade im Editor ist, gespeichert werden. D.h. es werden die Koordinaten aller Wände und Orientierungsmarkierungen in einer Textdatei gespeichert.

Die Textdatei kann mit Hilfe eines Texteditors editiert werden. Auf diese Weise kann man im Texteditor beliebig genaue Anordnungen der Wände und Orientierungsmarkierungen programmieren, was mit der Maus im Editor meistens nicht so genau hinzubekommen ist.



B.3.3 Markierungen löschen

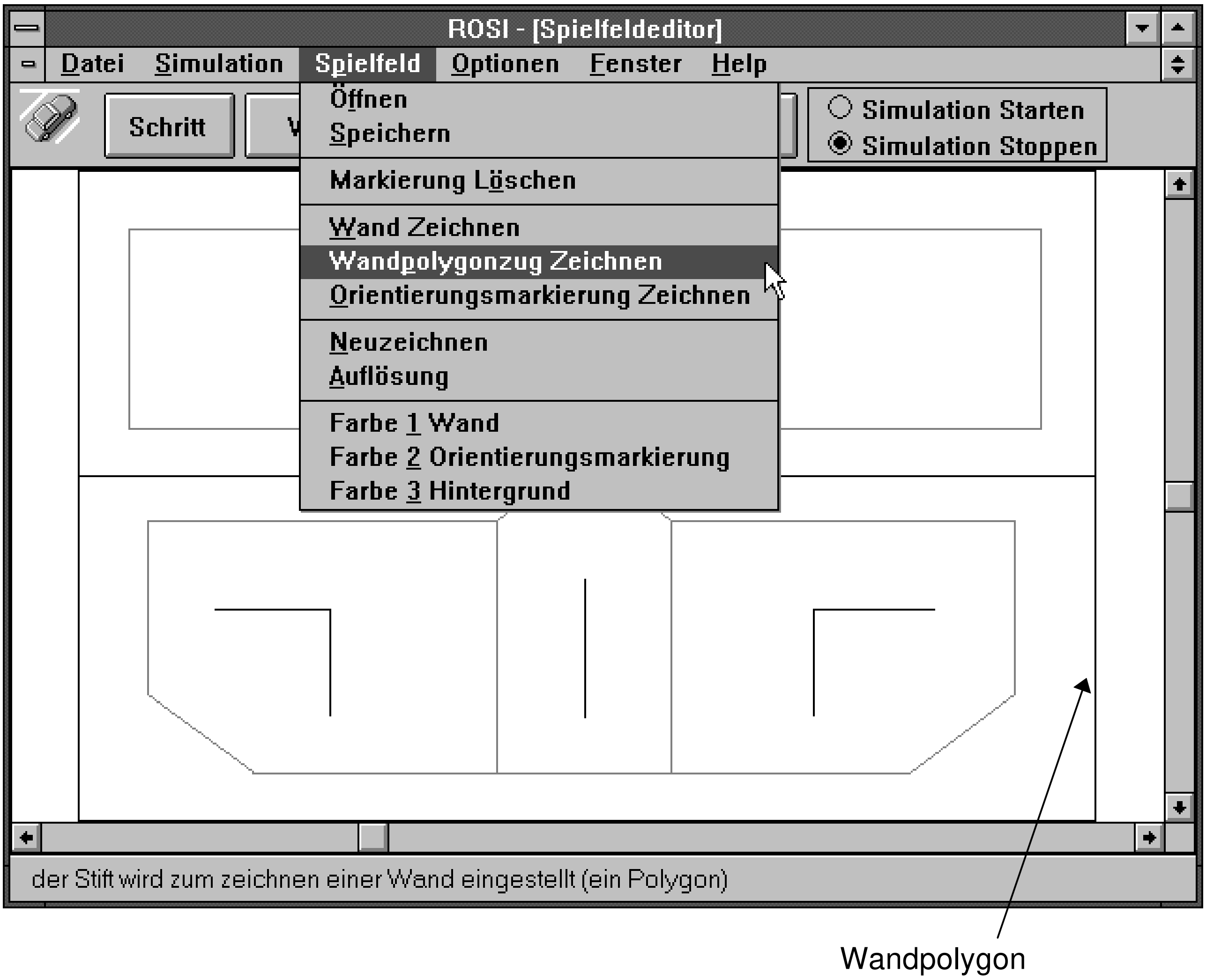
Wände und Orientierungsmarkierungen kann man markieren, indem man bei gedrückter Shift-Taste mit der linken Maustaste auf die entsprechende Wand bzw. die entsprechende Orientierungsmarkierung drückt. Ein markiertes Element erscheint rot und kann mit dem Menüpunkt Spielfeld / Markierung Löschen gelöscht werden.



B.3.4 Wand zeichnen

Der Mauscursor wird so eingestellt, daß eine Linie, die eine Wand darstellen soll, gezeichnet werden kann. Dazu muß nur die linke Maustaste an derjenigen Spielfeldposition gedrückt werden, wo sich das eine Wandende befinden soll. Die Maus ist zur Spielfeldposition zu bewegen, wo das andere Wandende sein soll und schließlich muß die linke Maustaste an diesem zweiten Wandende wieder losgelassen werden.

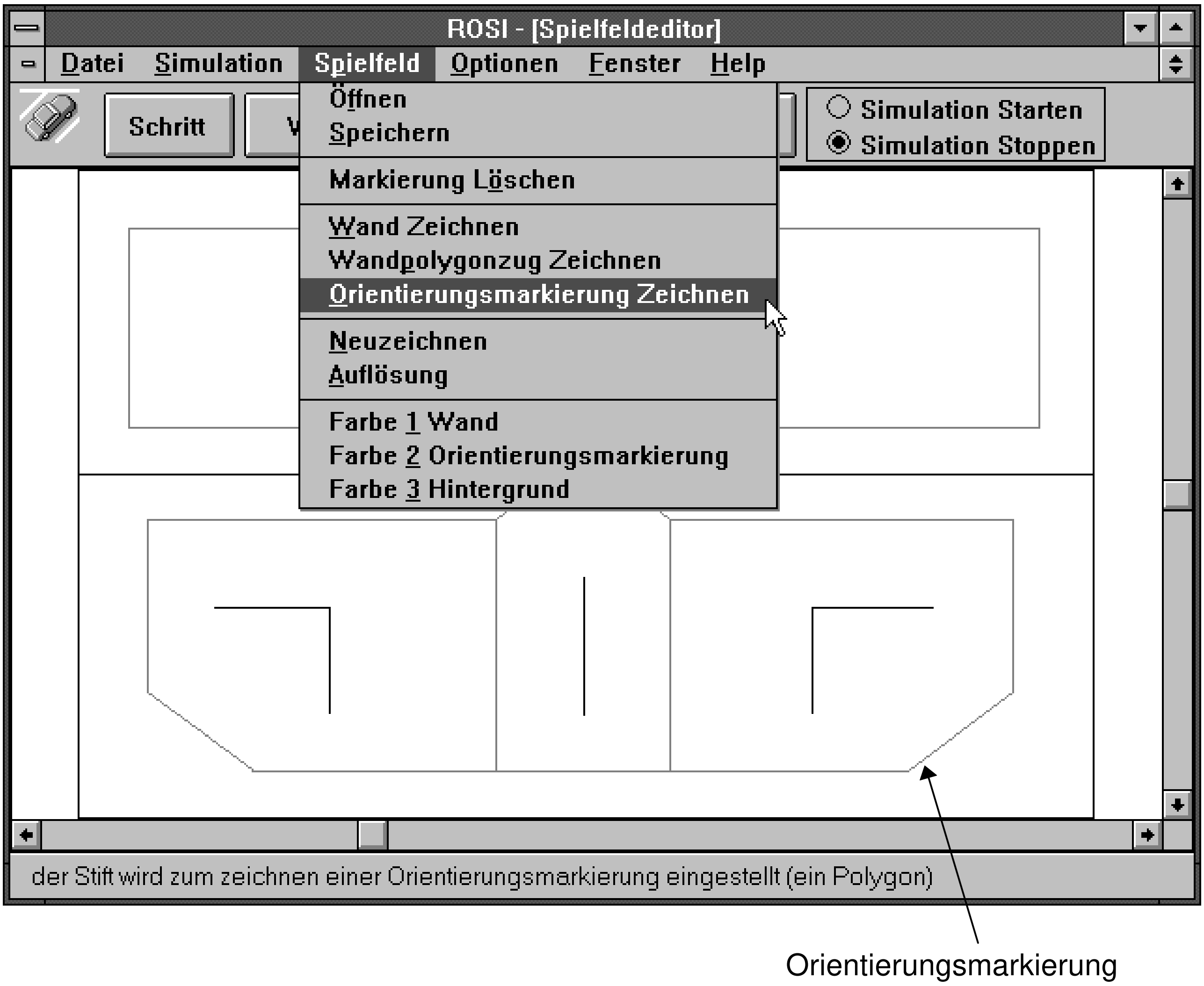
Während des zeichnens erscheint die noch nicht fertiggezeichnete Wand türkis.



B.3.5 Wandpolygon zeichnen

Der Mauscursor wird so eingestellt, daß ein Polygon gezeichnet werden kann, das eine Wand darstellen soll. Dazu drückt man die linke Maustaste und bewegt die Maus so, daß man eine türkise Polygonseite zeichnet; man läßt die Maustaste los, wenn die Seite fertiggezeichnet ist. Man kann die Maus nun zu beliebigen weiteren Punkten bewegen, die allesamt Eckpunkte des zu zeichnenden Polygons darstellen. An jedem Eckpunkt muß jeweils die linke Maustaste einmal gedrückt bzw. nur losgelassen werden, falls die Maustaste schon gedrückt war. Das Polygon wird abgeschlossen indem man zur linken Maustaste zusätzlich eine Shift-Taste drückt.

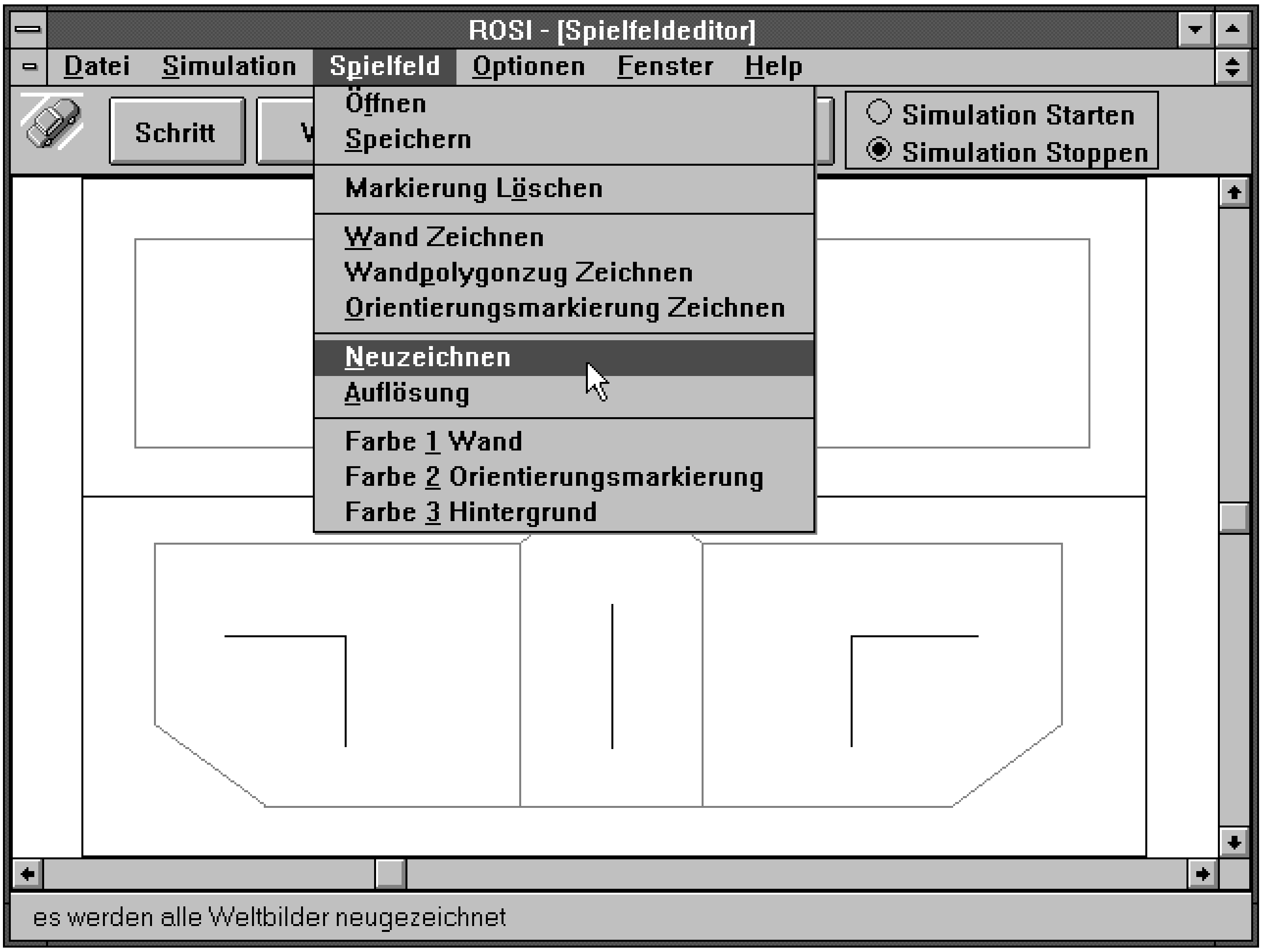
Während des zeichnens erscheint die noch nicht fertiggezeichnete Wand türkis.



B.3.6 Orientierungsmarkierung zeichnen

Der Mauscursor wird so eingestellt, daß ein Polygongezeichnet werden kann, das eine Orientierungsmarkierung darstellen soll. Dazu drückt man die linke Maustaste und bewegt die Maus so, daß man eine türkise Polygonseite zeichnet; man läßt die Maustaste los, wenn die Seite fertiggezeichnet ist. Man kann die Maus nun zu beliebigen weiteren Punkten bewegen, die allesamt Eckpunkte des zu zeichnenden Polygons darstellen. An jedem Eckpunkt muß jeweils die linke Maustaste einmal gedrückt bzw. nur losgelassen werden, falls die Maustaste schon gedrückt war. Das Polygon wird abgeschlossen indem man zur linken Maustaste zusätzlich eine Shift-Taste drückt.

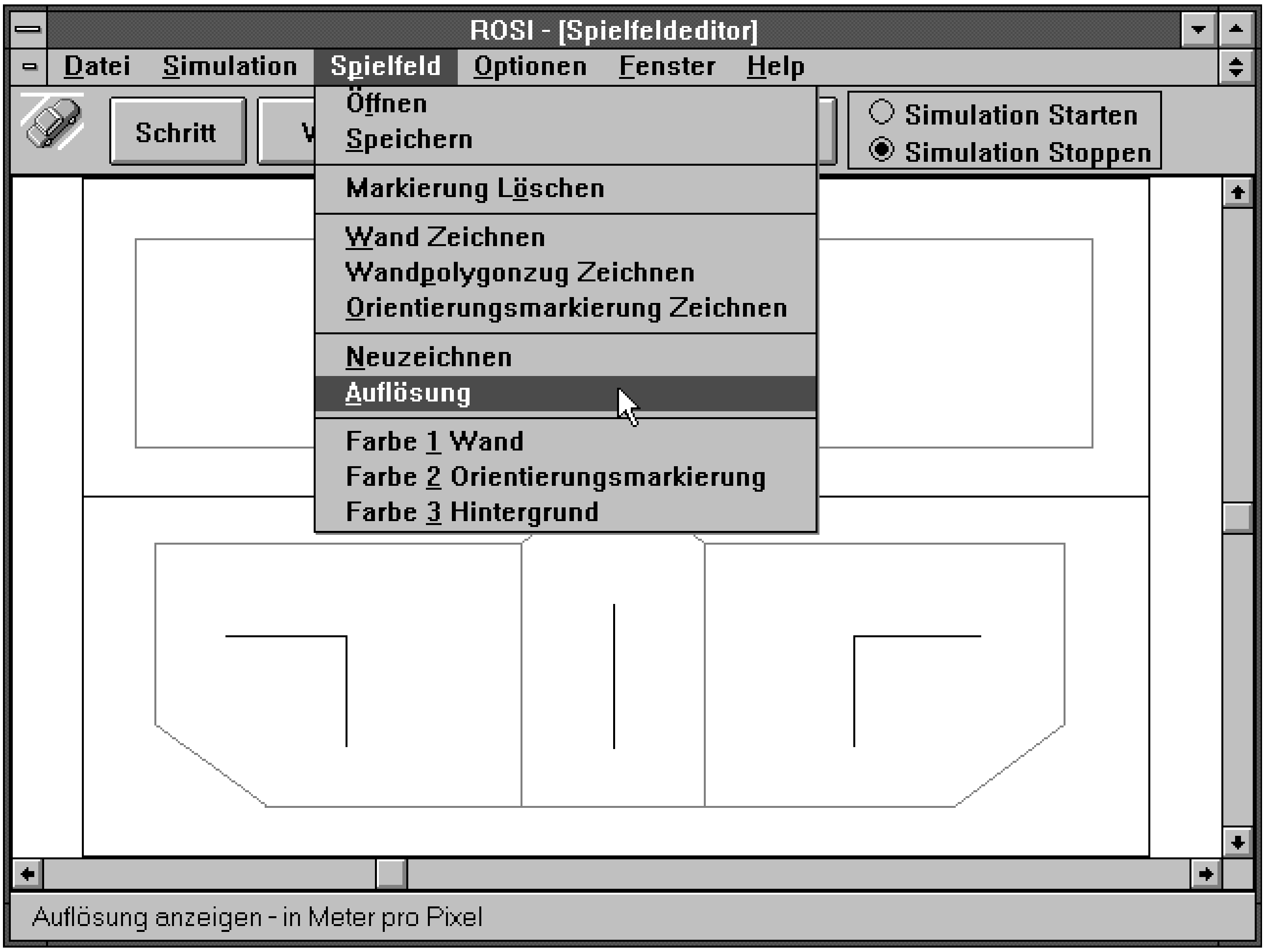
Während des zeichnens erscheint die noch nicht fertiggezeichnete Orientierungsmarkierung türkis.



B.3.7 Neuzeichnen

Mit diesem Menüpunkt wird lediglich alles einmal komplett neu gezeichnet.

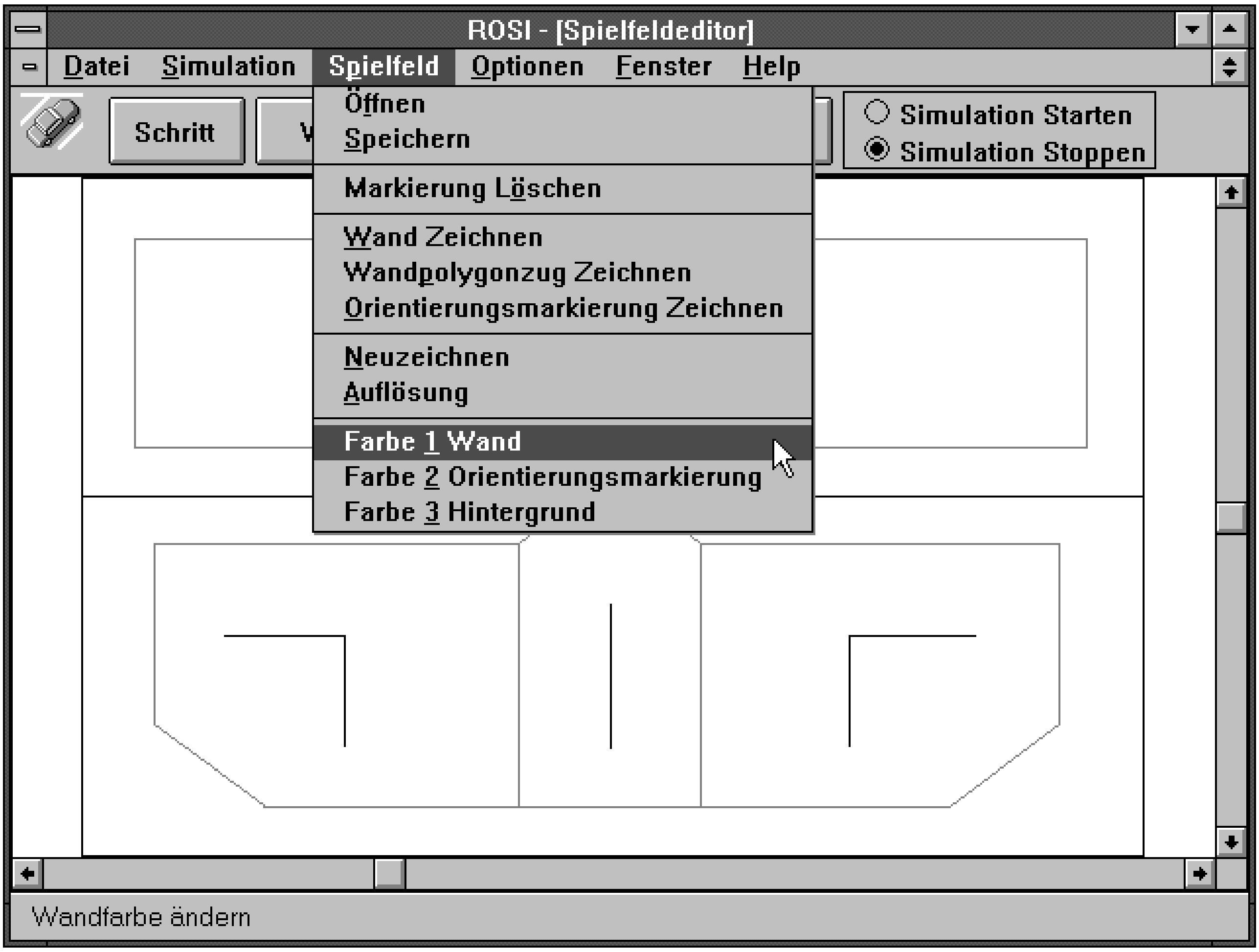
Das ist manchmal sinnvoll, falls irgendwelche graphischen Darstellungsprobleme auftreten, wie beispielsweise nicht korrekt wiederhergestellte Fenster. Es werden hierdurch keine Änderungen am Spielfeld vorgenommen.



B.3.8 Auflösung anzeigen

Mit diesem Menüpunkt wird die Darstellungsgenauigkeit angezeigt. Die Auflösung wird in Metern pro Pixel angegeben.

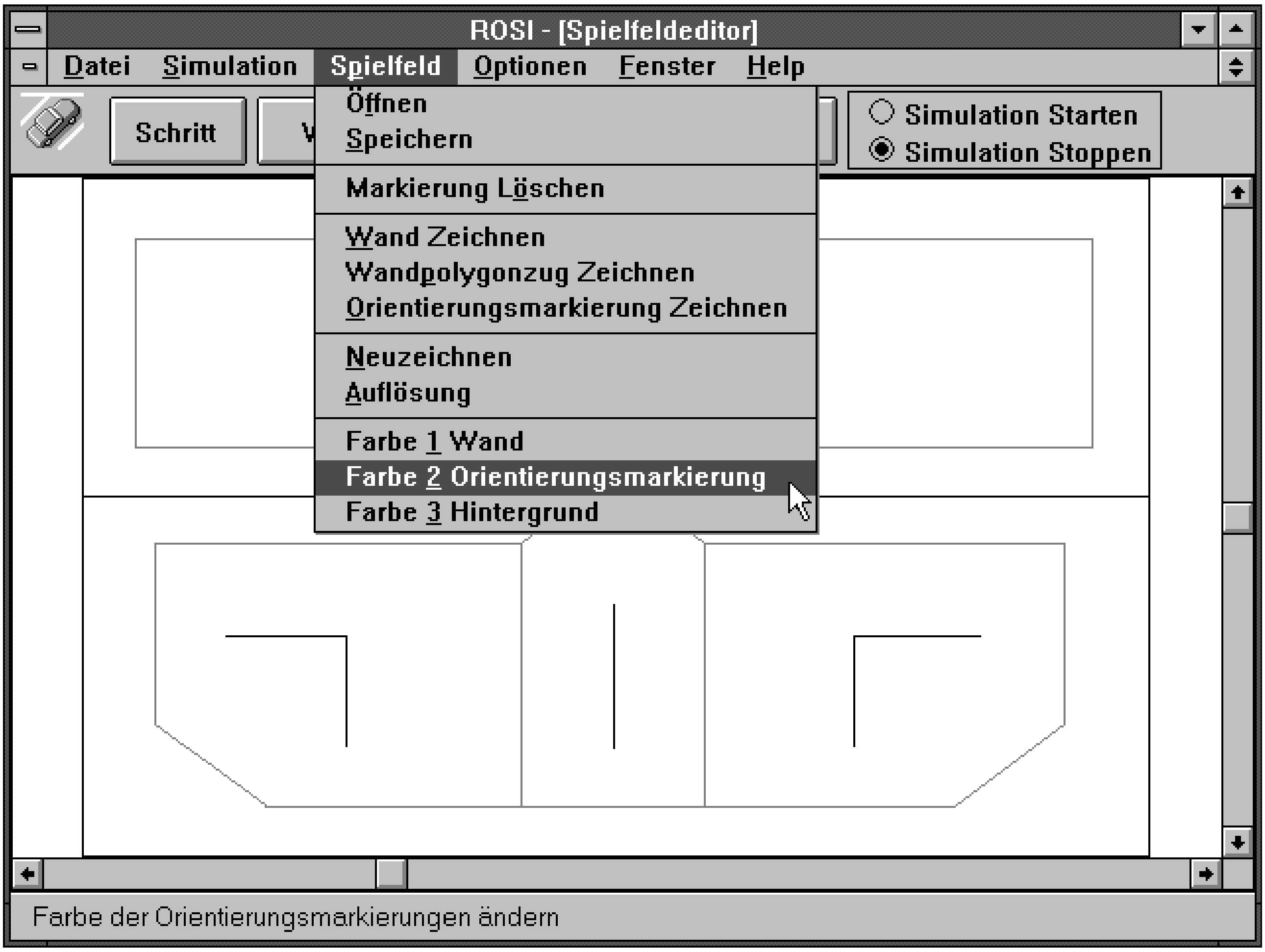
0,0058 Meter/Pixel bedeutet, daß auf ein Pixel ungefähr 5,8 Millimeter kommen.



B.3.9 Farbe der Wände festlegen

Mit diesem Menüpunkt wird die Farbe der Wände festgelegt.

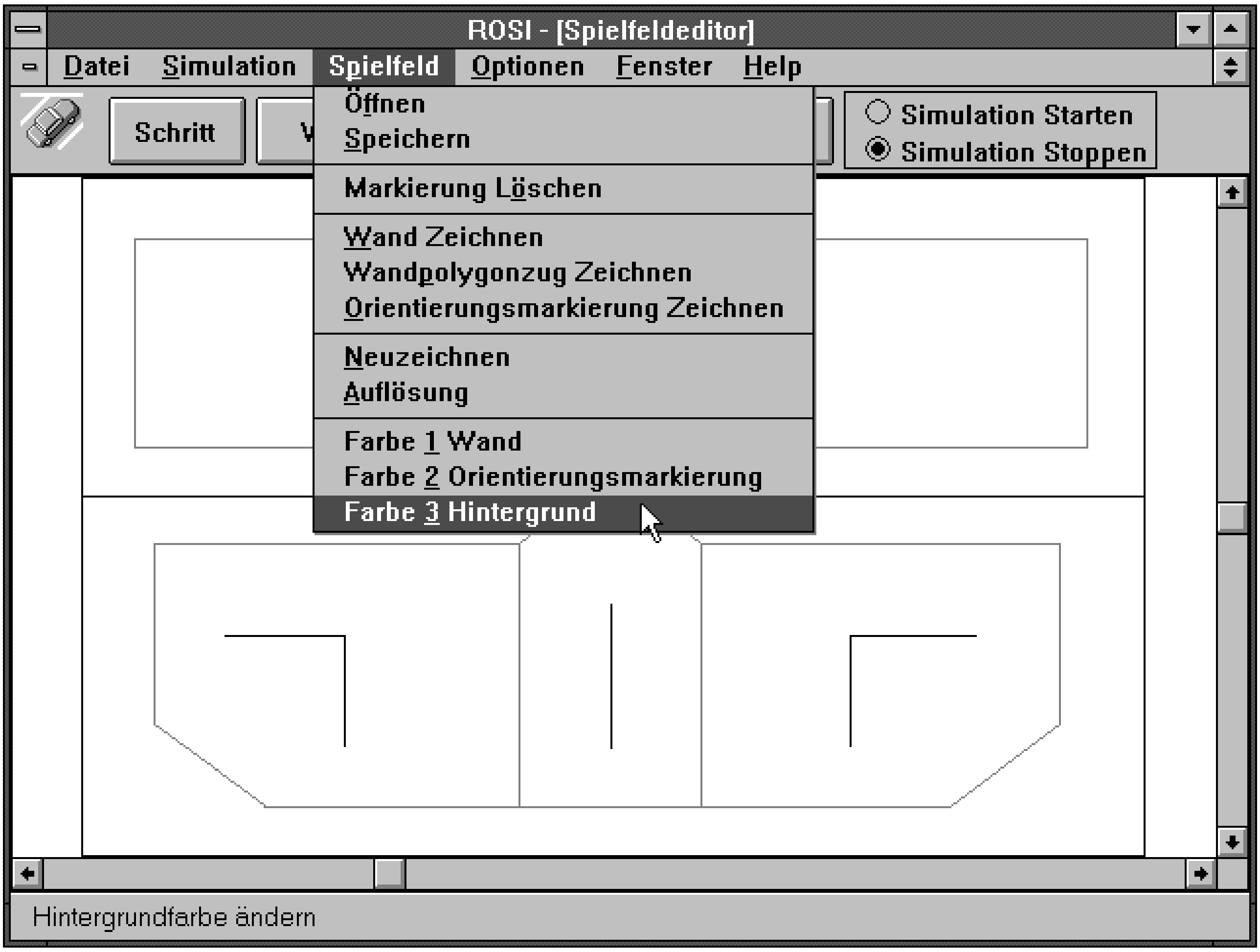
Es wird dazu der Windows gebräuliche Farbdialog benutzt.



B.3.10 Farbe der Orientierungsmarkierungen festlegen

Mit diesem Menüpunkt wird die Farbe der Orientierungsmarkierungen festgelegt.

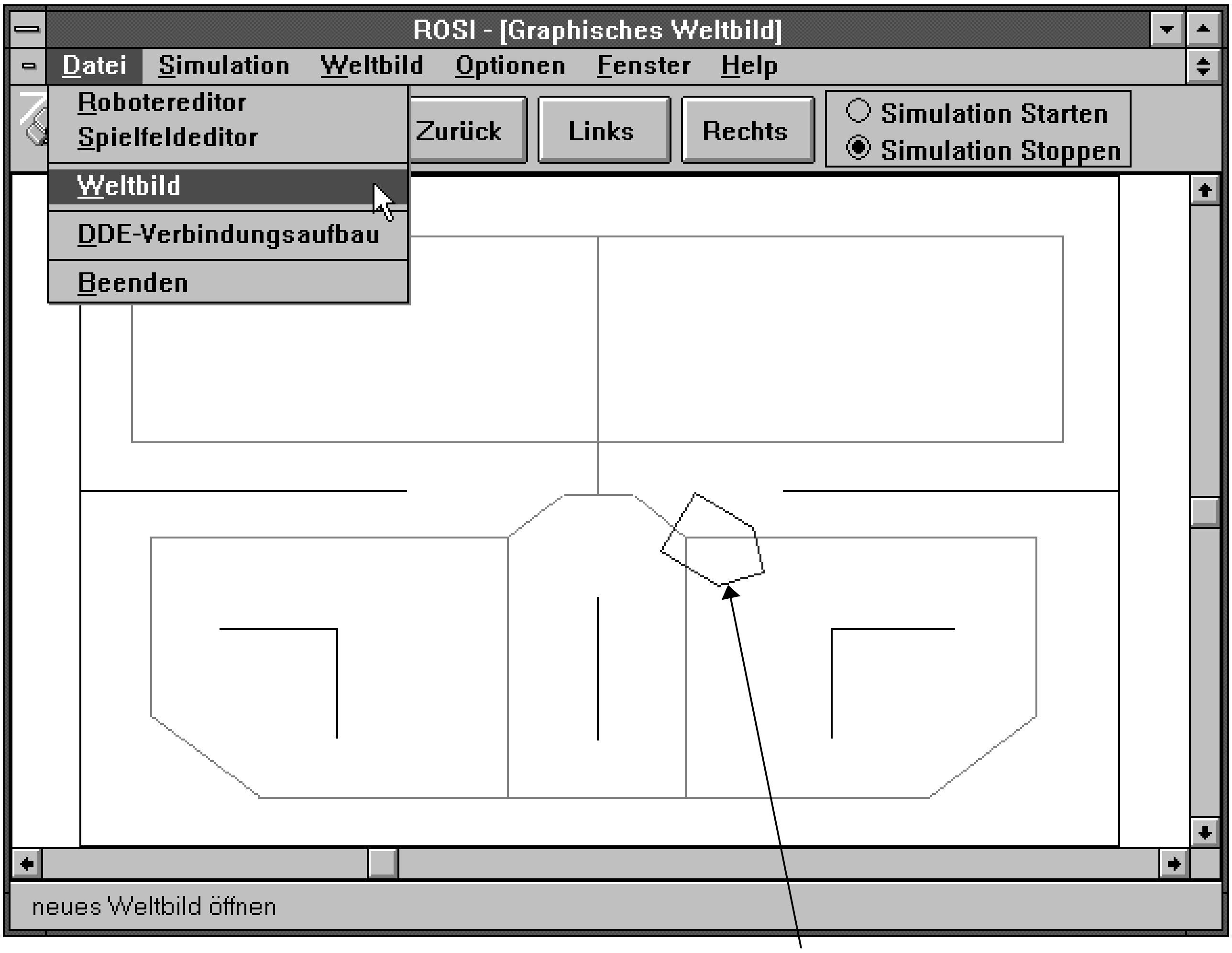
Es wird dazu der Windows gebräuliche Farbdialog benutzt.



B.3.11 Hintergrundfarbe des Spielfeldes festlegen

Mit diesem Menüpunkt wird die Hintergrundfarbe des Spielfeldes festgelegt.

Es wird dazu der Windows gebräuliche Farbdialog benutzt.



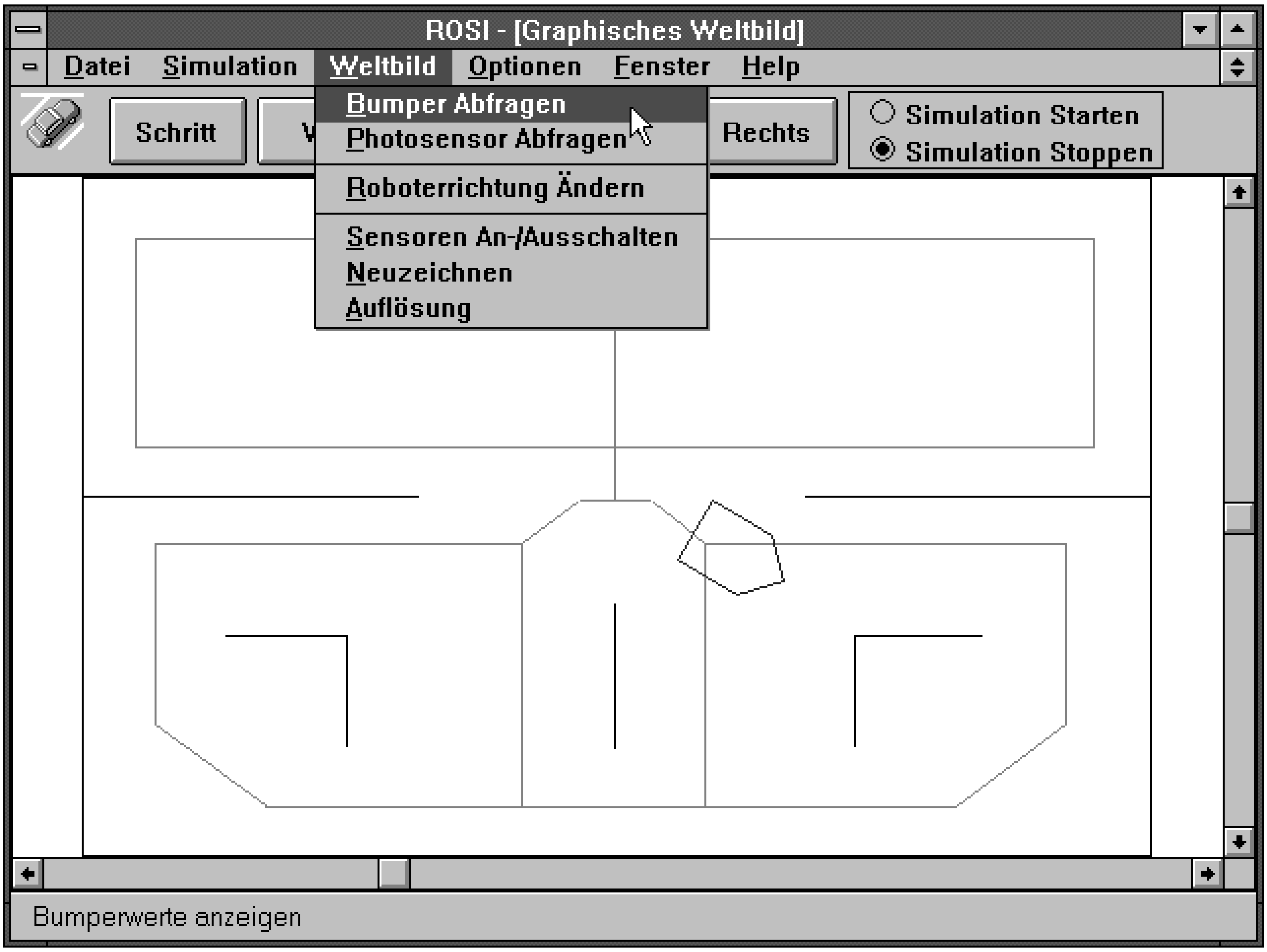
Der Roboter auf dem Spielfeld

B.4 Neues Weltbild öffnen

Über den Menüpunkt Datei / Weltbild wird ein Weltbild in einem neuen Fenster geöffnet. Dazu wird zunächst in einer Dialogbox nach der Darstellungsgenauigkeit gefragt. Je kleiner die angegebene Zahl ist, d.h. wieviel Meter ein Pixel darstellt, desto genauer ist die Auflösung der neuen Darstellung des Weltbildes.

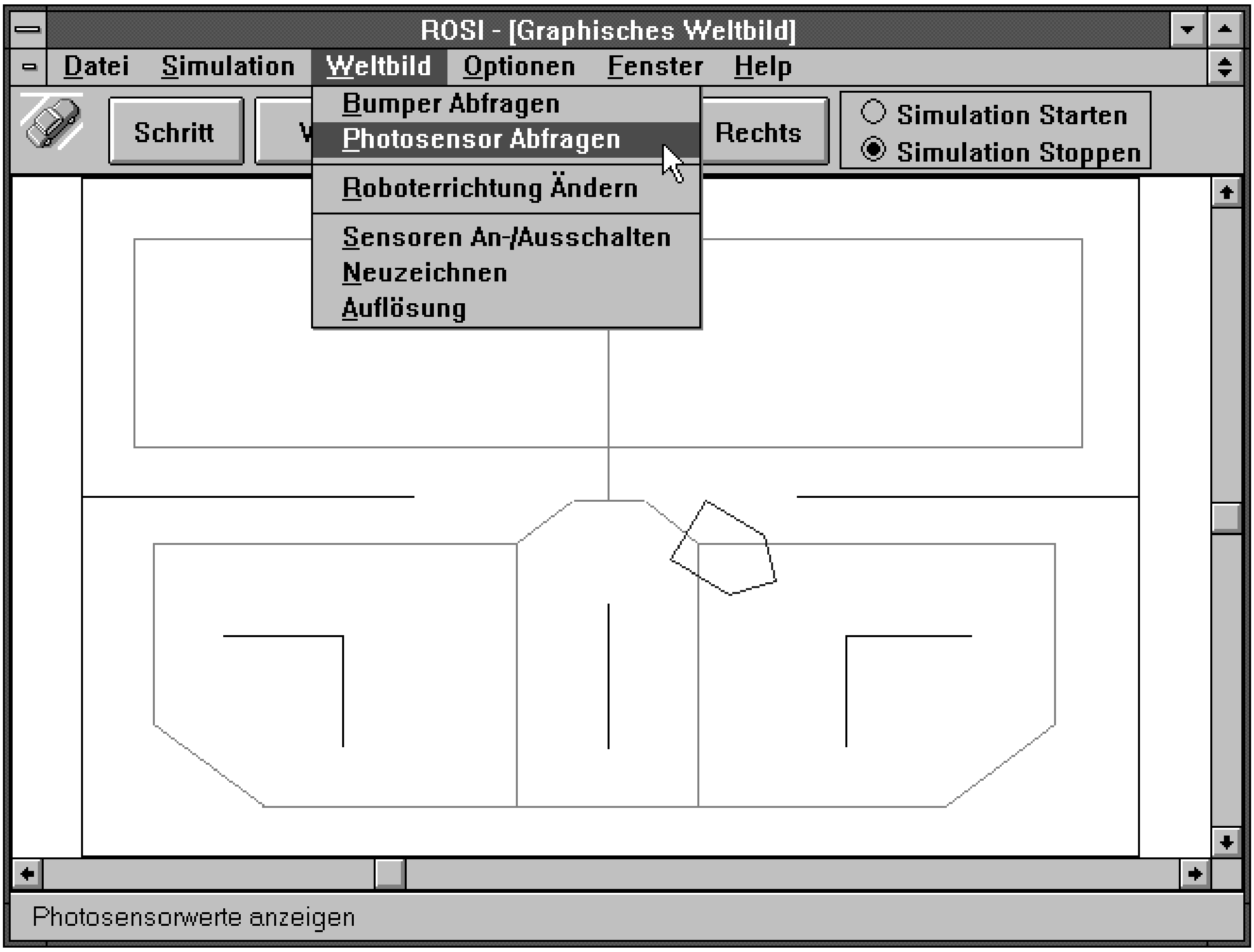
Es können parallel mehrere Weltbilder geöffnet werden. Auf diese Weise kann man z.B. unterschiedlich nahe herangezoomte Ausschnitte vergleichen.

Während im Robotereditor nur der Roboter für Konfigurationszwecke zu sehen ist und im Spielfeldeditor nur das Spielfeld, sieht man in einem Weltbild beides. Der Roboter befindet sich auf einer bestimmten Position auf dem Spielfeld.



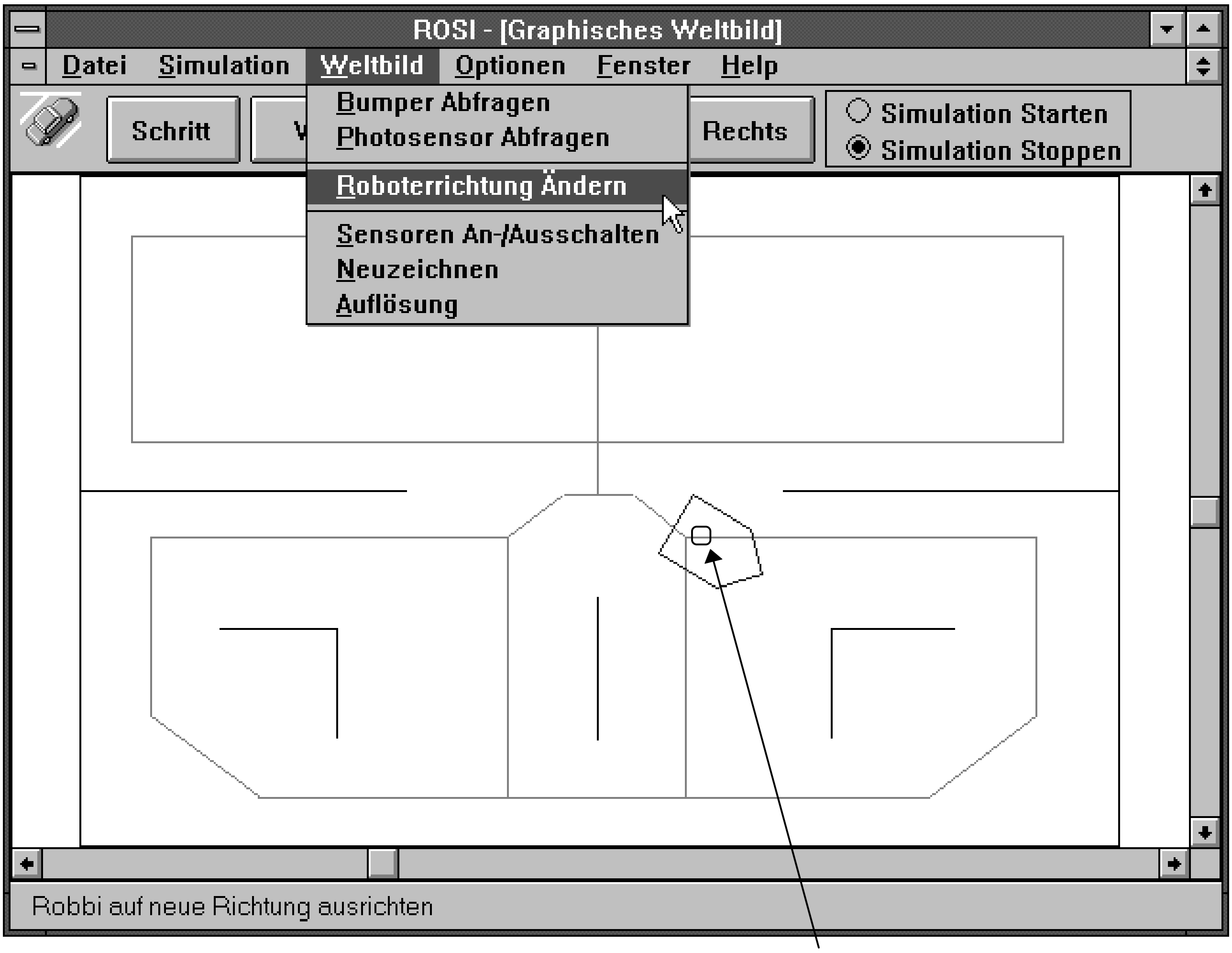
B.4.1 Mikroschalter abfragen

Mittels Weltbild / Bumper Abfragen können die Mikroschalter bzw. Bumper abgefragt werden. Dazu müssen deren Nummern angegeben werden. Die Sensornummern werden im Robotereditor automatisch vergeben und können dort auch abgefragt werden.



B.4.2 Photosensoren abfragen

Mittels Weltbild / Photosensor Abfragen können die Photosensorem abgefragt werden. Dazu müssen deren Nummern angegeben werden. Die Sensornummern werden im Robotereditor vergeben und können dort auch abgefragt werden.



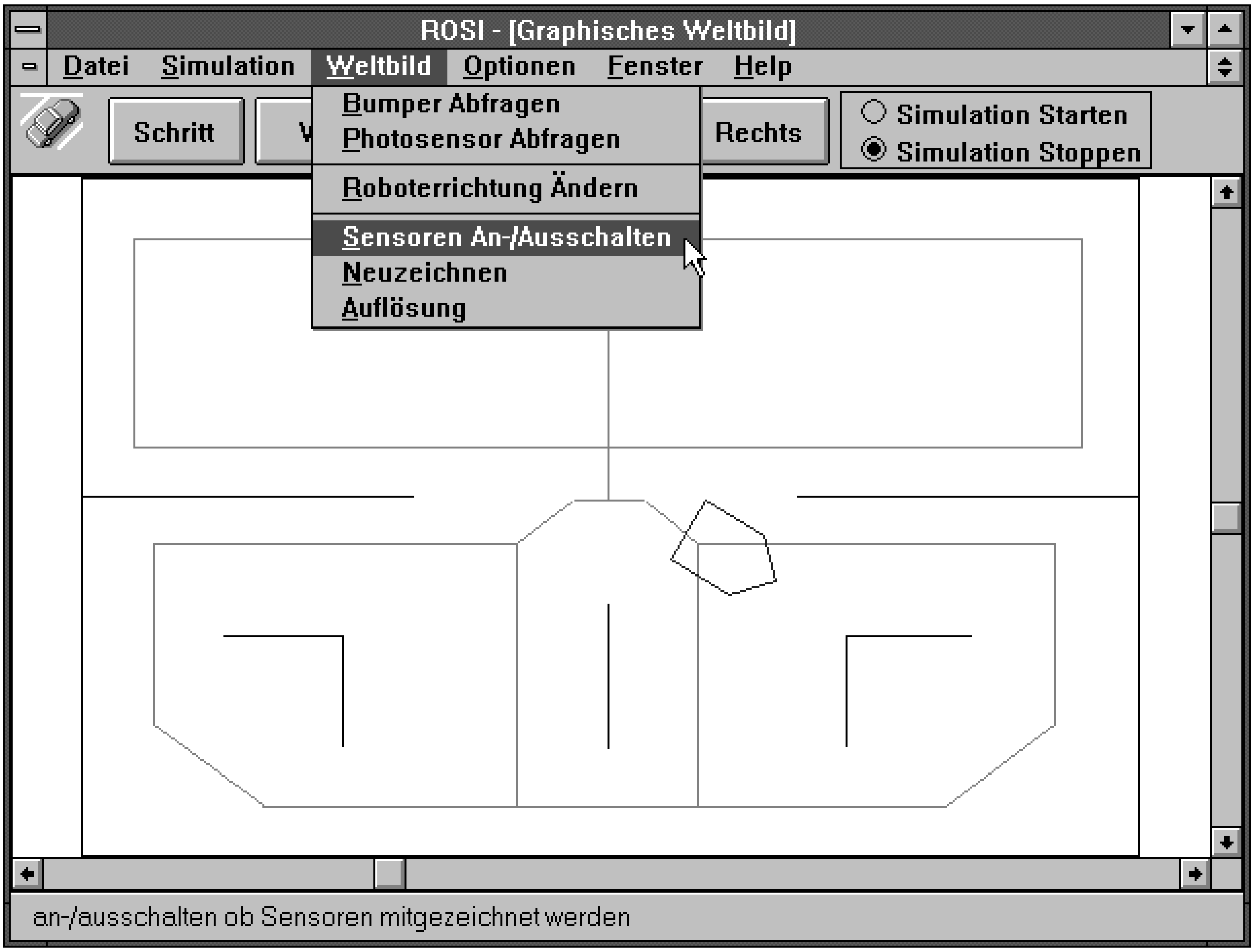
Photosensor der eine Orientierungsmarkierung erkennt

B.4.3 Richtung ändern

Durch den Menüpunkt Weltbild / Roboterrichtung ändern kann man die Roboterausrichtung relativ zum Spielfeld ändern. Bei der Angabe von 0° zeigt die Robotervorderseite nach oben. Bei der Angabe von 90° zeigt die Robotervorderseite nach rechts. Auf diese Weise läßt sich also eine bestimmte Richtung, in die der Roboter zeigt, angeben, um beispielsweise einen speziellen Zustand des Roboters auf dem Spielfeld einzustellen und zu testen.

B.4.4 Position ändern

Die Position des Roboters auf dem Spielfeld läßt sich ebenfalls ändern. Dazu muß man nur den Mauscursor auf die gewünschte Position plazieren und bei gedrückter Shift-Taste die linke Maustaste drücken. Das funktioniert aber nur, wenn die Simulationsschleife ROSIs ausgeschaltet ist. Sollte der Roboter auf die angewählte Position nicht passen, da eine Wand im Weg ist, so bleibt der Roboter auf seiner alten Position stehen.

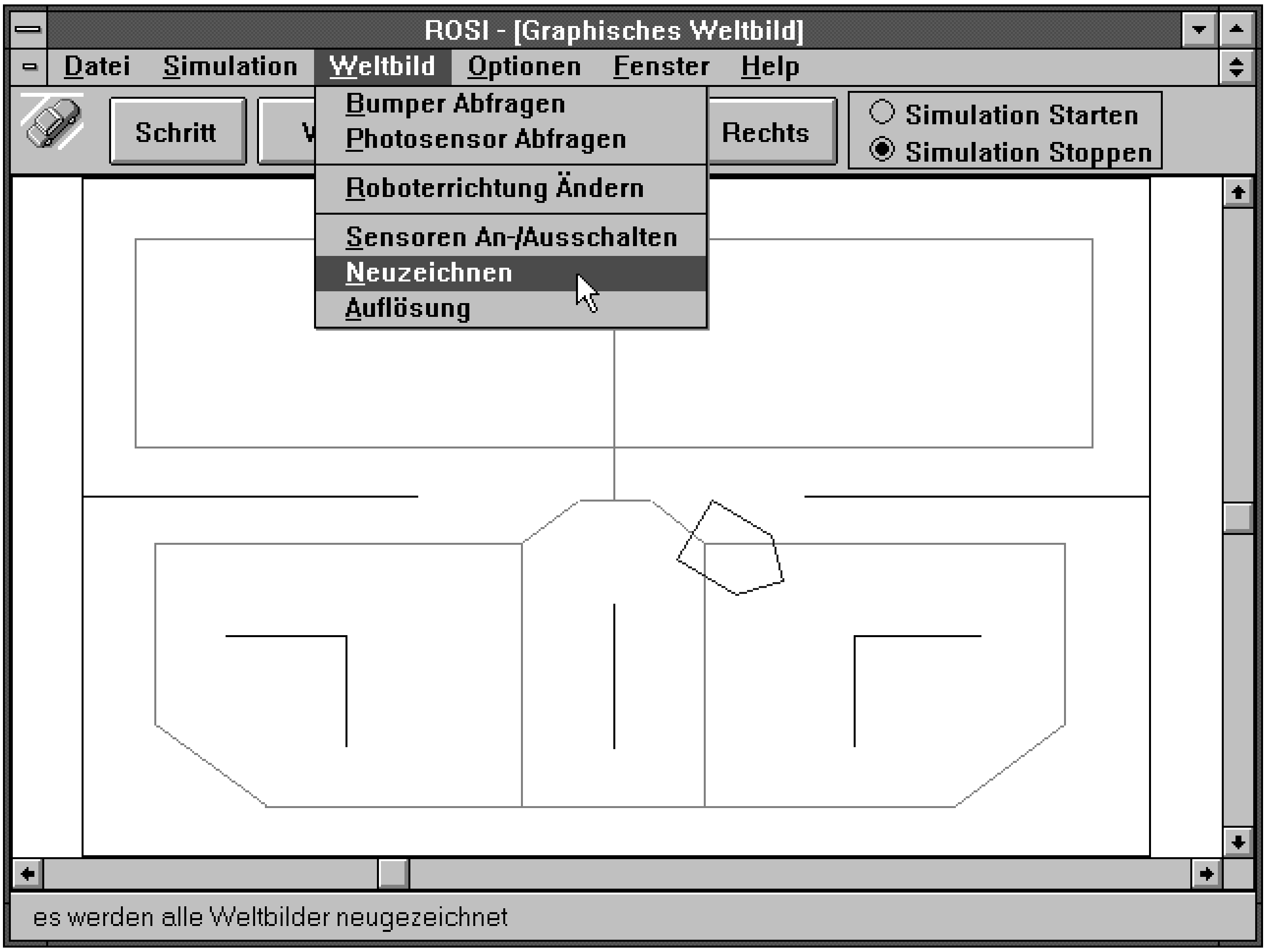


B.4.5 Sensoren anzeigen

Mit diesem Menüpunkt kann die Anzeige für die Sensoren ein- bzw. ausgeschaltet werden.

Während der Simulation können die Sensoren angezeigt werden. Das ist z.B. dann sinnvoll, wenn getestet werden soll, ob die Sensoren funktionieren. Wenn man beispielsweise einen Photosensor hat, der ein Rechteck darstellt, dann kann man sehen, ob im Rechteck eine Orientierungsmarkierung zu sehen ist. Entsprechend weiß man, ob die Werte richtig sind, die die Sensoren anzeigen.

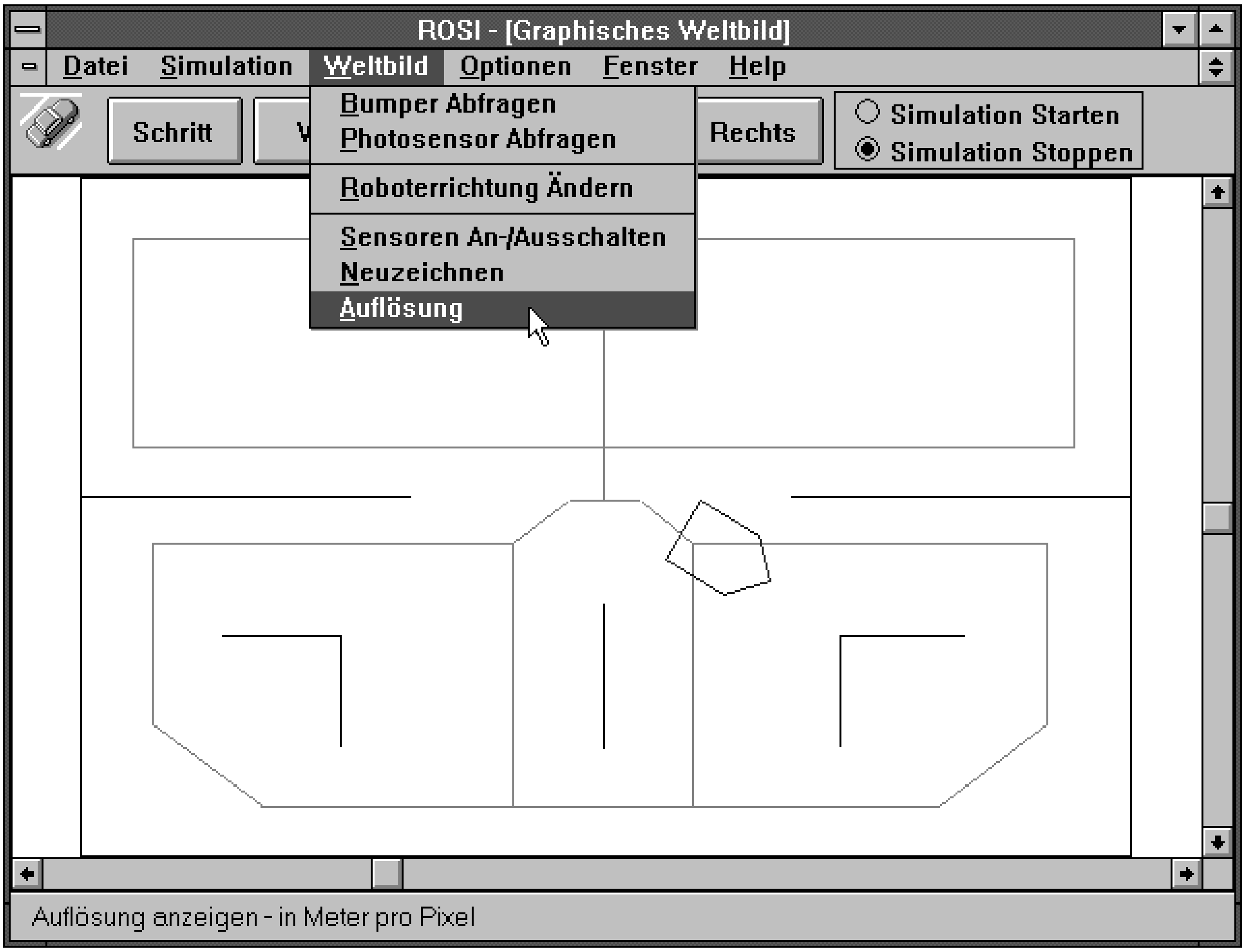
Die Anzeige hat keinen wesentlichen Einfluß auf die Simulationsgeschwindigkeit, da die Positionen der Sensoren sowieso immer für den Fall berechnet werden müssen, daß die Sensoren abgefragt werden.



B.4.6 Neuzeichnen

Mit diesem Menüpunkt wird lediglich alles einmal komplett neu gezeichnet.

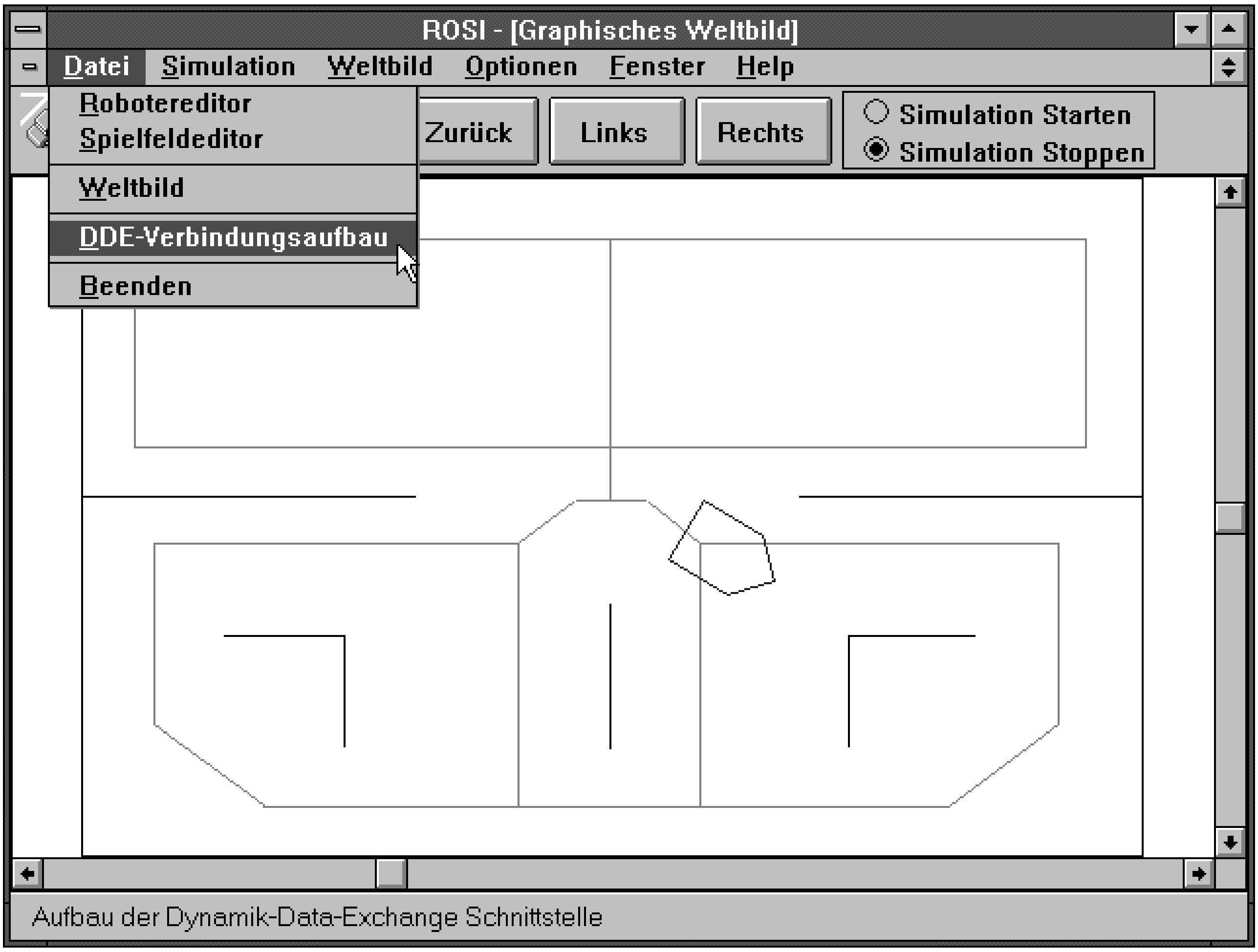
Das ist manchmal sinnvoll, falls irgendwelche graphischen Darstellungsprobleme auftreten, wie beispielsweise nicht korrekt wiederhergestellte Fenster. Hierdurch werden keine Änderungen vorgenommen.



B.4.7 Auflösung anzeigen

Mit diesem Menüpunkt wird die Darstellungsgenauigkeit angezeigt. Die Auflösung wird in Metern pro Pixel angegeben.

0,00075 Meter/Pixel bedeutet, daß auf ein Pixel ungefähr 0,00075 Meter kommen.



B.5 DDE-Verbindungsaufbau zur Simulatoransteuerung

Um eine DDE-Verbindung zu einem Steuerungsprogramm aufzubauen, ist dieser Menüpunkt zu wählen. Vorher muß allerdings das Steuerungsprogramm gestartet worden sein und es müssen alle DDE-Initialisierungen im Steuerungsprogramm vorgenommen worden sein. D.h. es ergibt sich folgende Reihenfolge beim DDE-Verbindungaufbau:

- ROSI starten
- Steuerungsprogramm starten und alle DDE-Initialisierungen durchführen
- Datei / DDE-Verbindungsaufbau aufrufen.

Es folgen vier Informationsfenster, die alle bloß bestätigt werden müssen. Sollte eines den Wert *nil* enthalten, dann ist der Verbindungsaufbau mißglückt. In diesem Fall sollten die Initialisierungsmethoden des Steuerungsprogamms für die DDE-Clients und DDE-Server überprüft werden. Sollte kein Informationsfenster den Wert *nil* enthalten, dann ist der Verbindungsaufbau geglückt.

Um die DDE-Schnittstelle zu nutzen, sind folgende Schritte notwendig:

- es ist eine DDE-Verbindung aufzubauen,

- es sind die DDE-Übertragungs-Methoden zu programmieren und

- die DDE-Verbindung ist wieder abzubauen.

Außerdem ist zu erklären, was alles berücksichtigt werden muß, wenn die DDE-Schnittstelle benutzt wird. Hierzu wird im folgenden ein Smalltalk-3.0.1-Beispiel zu Hilfe genommen. Um eine solche DDE-Schnittstelle in einer anderen Programmiersprache zu implementieren, müssen nur die Werte für *application*, *topic* und *class* bzw. *object* an diesem Beispiel abgeguckt werden.

Zunächst wird gezeigt, wie der Verbindungsaufbau der DDE-Schnittstelle in Smalltalk implementiert wird, anschließend wie die Übertragungsmethoden aussehen und schließlich wird der Verbindungsabbau erklärt.

B.5.1 DDE-Verbindungsaufbau

Bevor die DDE-Schnittstelle benutzt werden kann, müssen sogenannte Server- und Clientstellen eingerichtet werden. Soll beispielsweise vom Steuerungsprogramm ein Rotationsbefehl zum Simulator übertragen werden, dann muß im Steuerungsprogramm eine Serverstelle und im Simulator eine Clientstelle existieren. Zwischen Serverstelle und Clientstelle kann man sich einen Übertragungskanal denken, der Rotationsbefehle von der Serverstelle entgegennimmt und an die Clientstelle weiterleitet. Auf diese Weise können zwei voneinander unabhängige Windowsprogramme miteinander kommunizieren. In einem der zwei Programme muß lediglich eine Serverstelle und im anderen Programm lediglich eine dazu passende Clientstelle existieren.
Im Falle des Rotationsbefehls ist im Steuerungsprogramm eine Serverstelle einzurichten und im Simulator eine entsprechende Clientstelle. Die beiden Stellen werden miteinander verbunden, indem der sogenannte *application*-name derselbe ist und ebenso der *topic*-name. Als application-name bietet sich ROSI an und als topic-name beispielsweise *rotate*.
Im Falle des Translationsbefehls bietet sich der topic-name *translate* an, während der application-name derselbe bleibt.

Außerdem ist der Datentyp anzugeben, der entweder übertragen wird, der *object*-Wert, oder der empfangen wird, der *class*-Wert.
Bei den Rotations- und Translationsbefehlen handelt es sich um unidirektionale Übertragungen vom Steuerungsprogramm zum Simulator.

Bei den Sensorabfragen aber muß zunächst die Sensornummer vom Steuerungsprogramm zum Simulator übertragen werden und anschließend außerdem der Sensorwert vom Simulator zum Steuerungsprogramm. In diesem Fall hat man also einen bidirektionalen Übertragungskanal. Dieser läßt sich aber durch zwei unidirektionale Kanäle implementieren.

Es werden separat für beide Befehlsarten und für beide Sensorentypen eigene Kanäle eingerichtet. Eine Initialisierungsmethode für alle Server- und Clientstellen im Steuerungsprogramm könnte folgendermaßen aussehen:

```
initDDE
"PRIVATE - Es werden alle notwendigen DDE-Verbindungen zum Simulator 'ROSI' aufgebaut.
Damit der Verbindungsaufbau gelingt muß der Simulator bereits laufen."

"Bumper-Verbindung initialisieren"
bumperDDEClient:= DDEClient newClient: self application: 'ROSI' topic: 'BumperWert'.
bumperDDEClient notNil ifTrue:
    [bumperDDEClient hotLinkItem: 'BumperWert' class: Boolean selector: #item:BumperWert:].
bumperDDEServer:= DDEServer newServer: self application: 'ROSI' topic: 'BumperNr'.
bumperDDEServer notNil ifTrue:
    [bumperDDEServer addExportedItem: 'BumperNr' object: Integer].

"Photosensor-Verbindung initialisieren"
photoDDEClient:= DDEClient newClient: self application: 'ROSI' topic: 'PhotosensorWert'.
photoDDEClient notNil ifTrue:
   [photoDDEClient hotLinkItem: 'PhotosensorWert' class: Boolean selector: item:PhotosensorWert:].
photoDDEServer:= DDEServer newServer: self application: 'ROSI' topic: 'PhotosensorNr'.
photoDDEServer notNil ifTrue:
    [photoDDEServer addExportedItem: 'PhotosensorNr' object: Integer].

"Effektor-Verbindungen initialisieren"
rotateDDEServer:= DDEServer newServer: self application: 'ROSI' topic: 'rotate'.
rotateDDEServer notNil ifTrue: [rotateDDEServer addExportedItem: 'rotate' object: Float].
translateDDEServer:= DDEServer newServer: self application: 'ROSI' topic: 'translate'.
```

```
translateDDEServer notNil ifTrue: [translateDDEServer addExportedItem: 'translate' object: Float].
```

Für die Clientstellen müssen sogenannte *selector* angegeben werden. Das sind Methoden, die automatisch immer dann aufgerufen werden, wenn die Clientstelle von der Serverstelle des anderen Programms etwas übertragen bekommt.
Die Angabe das der Client ein *hotLinkItem* ist, bedeutet, daß automatisch an den Client ein Datenwert übertragen wird, sobald sich am Server der Wert dieses Datums ändert.

B.5.2 DDE-Übertragungsmethoden

Nachdem alle Server- und Clientstellen eingerichtet worden sind, können nun Daten übertragen werden. Um einen Photosensor abzufragen ist zunächst seine Nummer ROSI mitzuteilen:

```
photosensorNummerUebertragen: nr

"Es wird die Photosensornummer 'nr' an den Simulator 'ROSI' übertragen, damit der Simulator daraufhin den Wert dieses Photosensors zurückliefert. Gibt es den Photosensor mit dieser Nummer 'nr' nicht, dann wird '0' für den Wert dieses Sensors zurückgeliefert."

photoDDEServer updateExportedItem: 'PhotosensorNr' object: nr!
```

Der Photosensorwert wird von ROSI übertragen und löst daraufhin im Steuerungsprogramm die folgende Methode aus, die den Wert in einer Variable photoWert speichert:

```
item: i PhotosensorWert: pw

"PRIVATE - Diese Methode wird automatisch aufgerufen, wenn 'photosensorDDEClient' die Nachricht eines neuen Photosensorwertes erhält. Hierbei handelt es sich um denjenigen Photosensor, dessen Nummer zuvor mittels 'photosensorNummerUebertragen: nr' übertragen wurde."

photoWert:= pw.!
```

Entsprechend funktioniert die Abfrage der Mikroschalterwerte.

Die Übertragungen der Rotations- und Translationsbefehle sind noch einfacher, da die Kommunikation nur unidirektional vom Steuerungsprogramm zu ROSI verläuft:

```
rotate: grad

"An den Simulator 'ROSI' übertragen, wie weit Robbi rotieren soll (links = negativ)."

rotateDDEServer updateExportedItem: 'rotate' object: grad!
```

```
translate: meter

"An den Simulator 'ROSI' übertragen, wie weit Robbi translatieren soll (rückwärts = negativ)."

translateDDEServer updateExportedItem: 'translate' object: meter!
```

Sind beispielsweise alle diese DDE-Methoden in einer Klasse *Steuerungsebene0* implementiert, dann könnte man von dieser Klasse eine mit dem Namen *Steuerungsebene1* ableiten. Die *Steuerungsebene1* würde aus den Steuerungsprimitiven der Klasse *Steuerungsebene0* komplexere Funktionen zusammenbauen.
Die *Steuerungsebene0* enthält die Methoden zur Rotation, Translation, Mikroschalterabfrage und Photosensorabfrage. Die *Steuerungsebene1* kann diese Methoden in beliebiger Reihenfolge und in beliebigem Umfang benutzen. Ebenso könnte eine weitere *Steuerungsebene2* auf den Methoden der *Steuerungsebene1* aufbauen usw..

Eine Methode, die den Roboter zufällig k Schritte auf dem Spielfeld hin und her laufen lassen soll, könnte auf der der *Steuerungsebene1* folgendermaßen aussehen:

```
kZufallsSchritteMachen: k


|zaehler zufallsZahl|
```

```
zaehler:= 0.
(zaehler < k) whileTrue:
[
    zufallsZahl:= Random between1and: 4.
    zufallsZahl = 1 ifTrue:[self rotate: 0.1]. „um 0.1° drehen"
    zufallsZahl = 2 ifTrue:[self translate: 0.01]. „um 0.01 m = 1 cm geradeausbewegen"
    zufallsZahl = 3 ifTrue:[self rotate: -0.1]. „um -0.1° drehen"
    zufallsZahl = 4 ifTrue:[self translate: -0.01]. „um -0.01 m = 1 cm rückwärtsbewegen"
    zaehler:= zaehler + 1.
]
```

Eine Methode, nach welcher der Roboter sich solange hin und her bewegen würde, bis er eine Orientierungsmarkierung erkennt, könnte folgendermaßen aussehen:

```
sucheOrientierungsmarkierung: k


|zufallsZahl|

(photosensor: 1 ist: 0) whileTrue:
[
    zufallsZahl:= Random between1and: 4.
    zufallsZahl = 1 ifTrue:[self rotate: 0.1]. „um 0.1° drehen"
    zufallsZahl = 2 ifTrue:[self translate: 0.01]. „um 0.01 m = 1 cm geradeausbewegen"
    zufallsZahl = 3 ifTrue:[self rotate: -0.1]. „um -0.1° drehen"
    zufallsZahl = 4 ifTrue:[self translate: -0.01]. „um -0.01 m = 1 cm rückwärtsbewegen"
]
```

Die Methode *photosensor: 1 ist: 0* soll hierbei die Photosensorabfrage kapseln. D.h. in dieser Methode muß zunächst die Sensornummer 1 übertragen werden, eine Weile gewartet und schließlich der Photosensorwert abgefragt werden.
Diese Methode könnte folgendermaßen aussehen:

```
photosensor: i ist: k
```

```
|s0|

s0:= Steuerungsebene0 new. „Instanz der Ebene 0 erzeugen“
s0 photosensorNummerUebertragen: i. „an Simulator eine Sensoranfrage schicken“
s0 einschlafen. „mittels Semaphore warten bis Simulator fertig ist“
^(s0 photoWert = k). „Wert abfragen und Ergebnis zurückliefern“
```

B.5.3 DDE-Verbindungsabbau

Um Speicher, der für die Übertragungskanäle benötigt wurde, und andere Einstellungen im Betriebssystem zur Beendigung der DDE-Verbindungen vorzunehmen, müssen alle Kanäle wieder abgebaut werden. Die Methode zum Beenden der DDE-Verbindungen sieht folgendermaßen aus:

```
close
"Diese Methode ist bei Zerstörung dieses Objekts 'self' aufzurufen, damit alle DDE-Verbindungen aufgelöst werden."

bumperDDEClient notNil ifTrue: [bumperDDEClient terminate. bumperDDEClient:= nil].
bumperDDEServer notNil ifTrue: [bumperDDEServer terminate. bumperDDEServer:= nil].
photoDDEClient notNil ifTrue: [photoDDEClient terminate. photoDDEClient:= nil].
photoDDEServer notNil ifTrue: [photoDDEServer terminate. photoDDEServer:= nil].
rotateDDEServer notNil ifTrue: [rotateDDEServer terminate. rotateDDEServer:= nil].
translateDDEServer notNil ifTrue: [translateDDEServer terminate. translateDDEServer:= nil].!
```

Es müssen also nur alle Server- und Clientstellen terminiert und die Variable auf nil gesetzt werden.

B.5.4 DDE-Einsatzvorgehensweise

Um die DDE-Verbindung herzustellen ist

- ROSI zu starten, womit die Serverstellen ROSIs initialisiert wären,

- das Steuerungsprogramm zu starten,

- die Serverstellen und Clientstellen im Steuerungsprogramm zu initialisieren (initDDE) und

- schließlich ist im ROSI-Menü DATEI/DDE-Verbindungsaufbau aufzurufen, um die Clientstellen ROSIs zu initialisieren; hierbei sind die im folgenden auf dem Bildschirm erscheinenden vier Informationsfenster zu bestätigen;
  ein mit *nil* angezeigtes Fenster zeigt, daß der Verbindungsaufbau mißlungen ist.

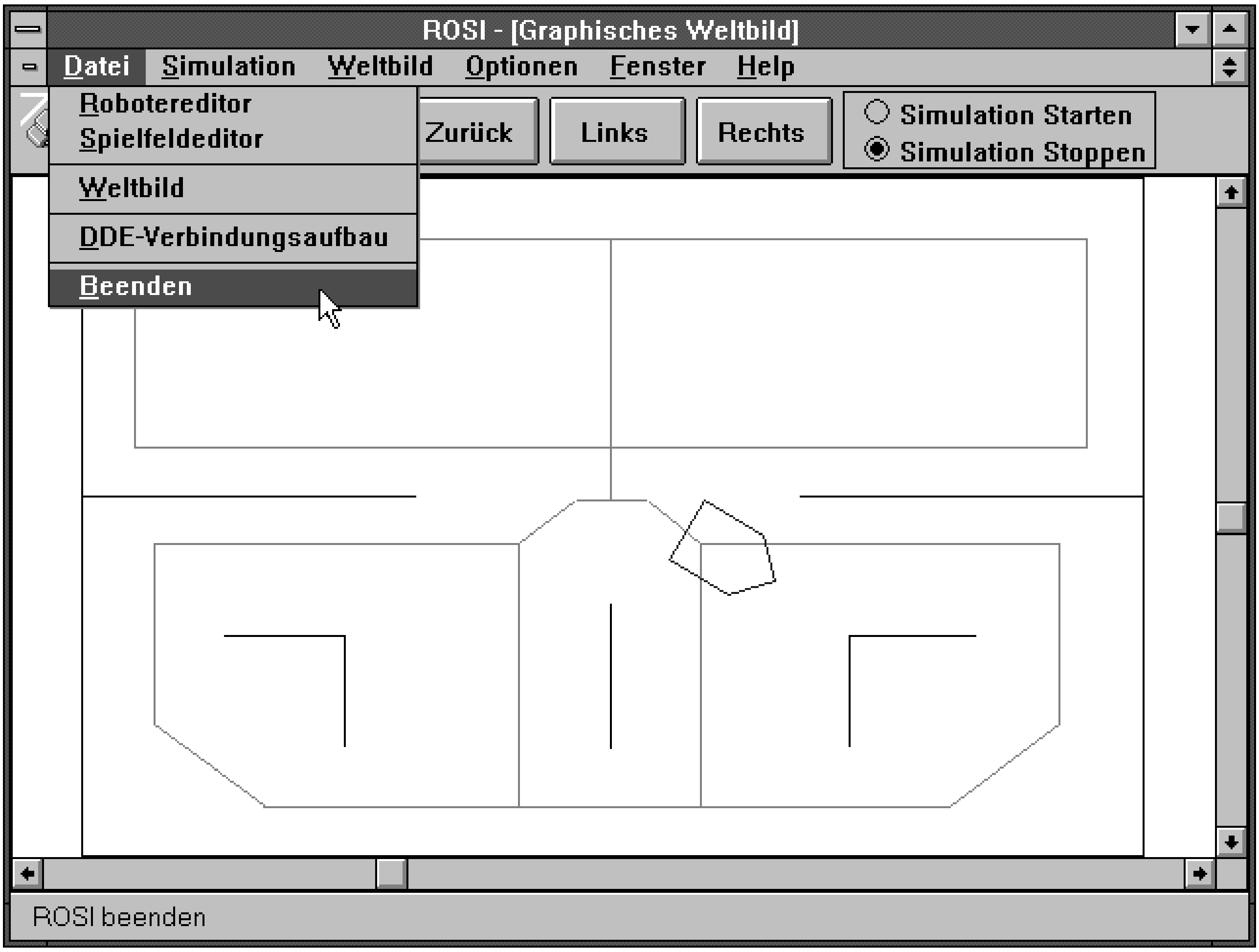
Diese Vorgehensweise ist deshalb notwendig, da Server vor Clients initialisiert werden müssen. Zu jedem Server im Steuerungsprogramm gibt es einen Client bei ROSI und umgekehrt. Die Server ROSIs werden beim Start ROSIs eingerichtet, so daß anschließend im Steuerungsprogramm sowohl alle Server als auch alle Clients eingerichtet werden können. Schließlich müssen noch die Clients ROSIs eingerichtet werden, die zu den Servern des Steuerungsprogramms gehören.

Es ist zu beachten, daß bei der Übertragung der Werte über die DDE-Schnittstelle Verzögerungen einzukalkulieren sind, da die Übertragung über das Betriebssystem Windows geht und somit abhängig von allen zur Verfügung stehenden Ressourcen ist. Das betrifft insbesondere die Zeit.

Aus diesem Grund muß darauf geachtet werden, daß nach Übertragung einer Sensornummer gewartet wird, bis der Sensorwert abgefragt werden kann. Würde man zu früh den Sensorwert abfragen, etwa nachdem man die Sensornummer gerade erst übertragen hat, kann es passieren, daß man noch einen alten Sensorwert vorfindet.

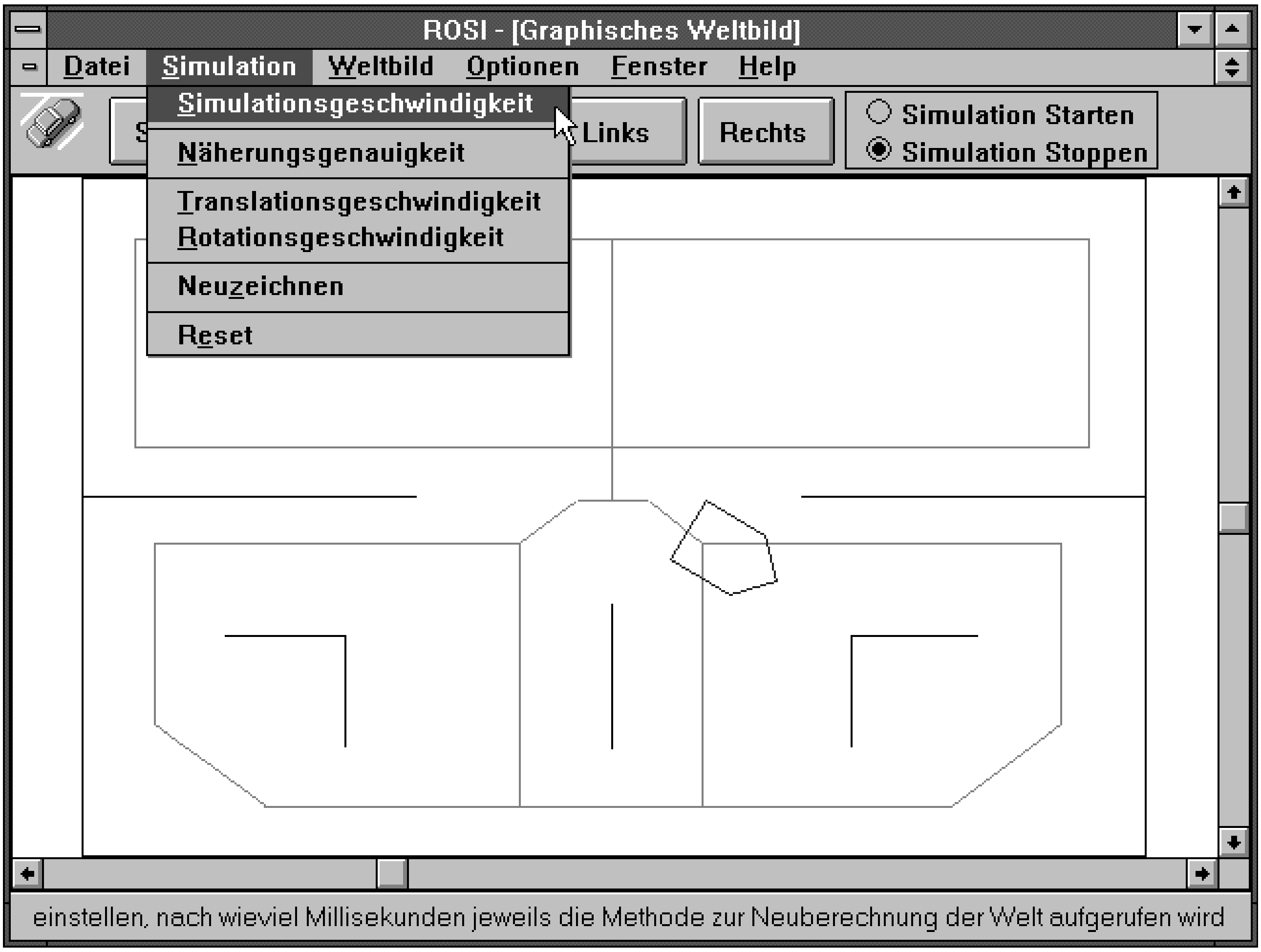
Zum einen dauert die Übertragung, welcher Sensor abzufragen ist, eine Weile, zum anderen die Ermittlung ROSIs, was dieser Sensor erkennt. Und schließlich muß dieser erkannte Wert von ROSI wieder Rückübertragen werden.

Deshalb ist eine künstliche Verzögerung im Steuerungsprogramm nötig. Diese sorgt dann dafür, daß genug Zeit verstreicht, ehe das Steuerungsprogramm den Sensorwert benutzt. Eine solche Verzögerung kann am besten mit Hilfe von Semaphoren, die das Steuerungsprogramm schlafen legen bis der Sensorwert vorliegt, implementiert werden.



B.6 ROSI beenden

Mit diesem Menüpunkt wird der Simulator beendet.



B.7 Simulationseinstellungen

Während mit dem Robotereditor und dem Spielfeldeditor statische Aspekte des Roboters und des Spielfeldes konfiguriert werden, können mit den Simulationseinstellungen dynamische Aspekte eingestellt werden, die während der Simulation eine Rolle spielen: die Geschwindigkeit der Simulationsausführung, d.h. eine zeitlupenähnliche oder schnellere Simulation, die Angabe der Näherungsgenauigkeit und die Geschwindigkeit des Roboters, d.h. genauer die Angabe seiner Translations- und Rotationsschrittgrößen.

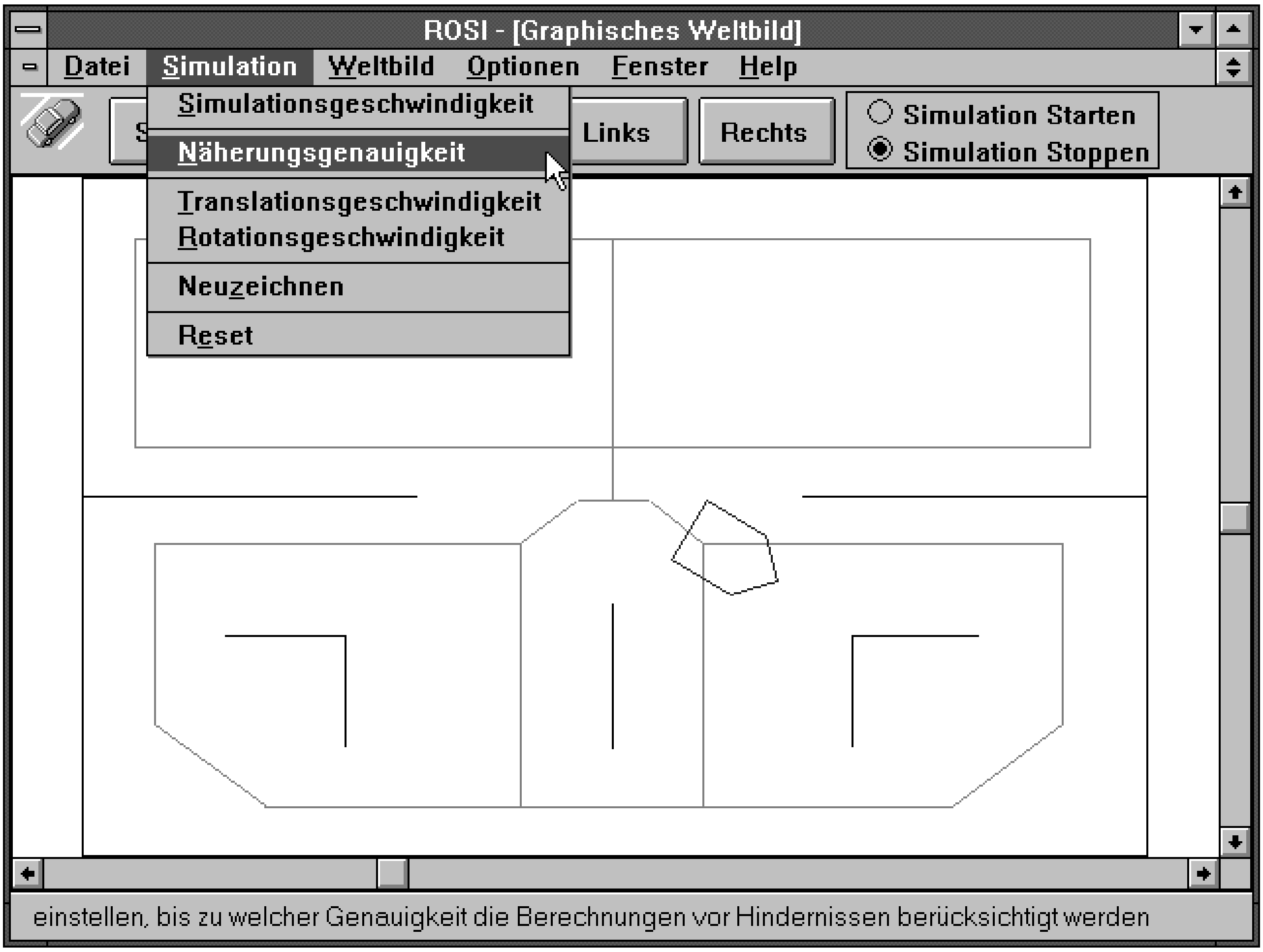
B.7.1 Simulationsgeschwindigkeit

Die Simulationsgeschwindigkeit gibt an, nach wievielen Millisekunden der nächste Simulationsschritt jeweils berechnet wird.

Mit einer hohen Angabe für die Simulationsgeschwindigkeit erreicht man eine Zeitlupendarstellung der Simulation.

Um eine möglichst flüssige Anzeige der Simulation zu bekommen, muß man einen kleinen Wert angeben.

Sollte der Wert unterhalb der möglichen Geschwindigkeit des Computers liegen, dann wird der niedrigsmögliche Wert automatisch ermittelt und benutzt. Aus diesem Grund sollte man einfach den Wert 0 eingeben, wenn man die schnellstmögliche Simulationsgeschwindigkeit mutzen möchte.

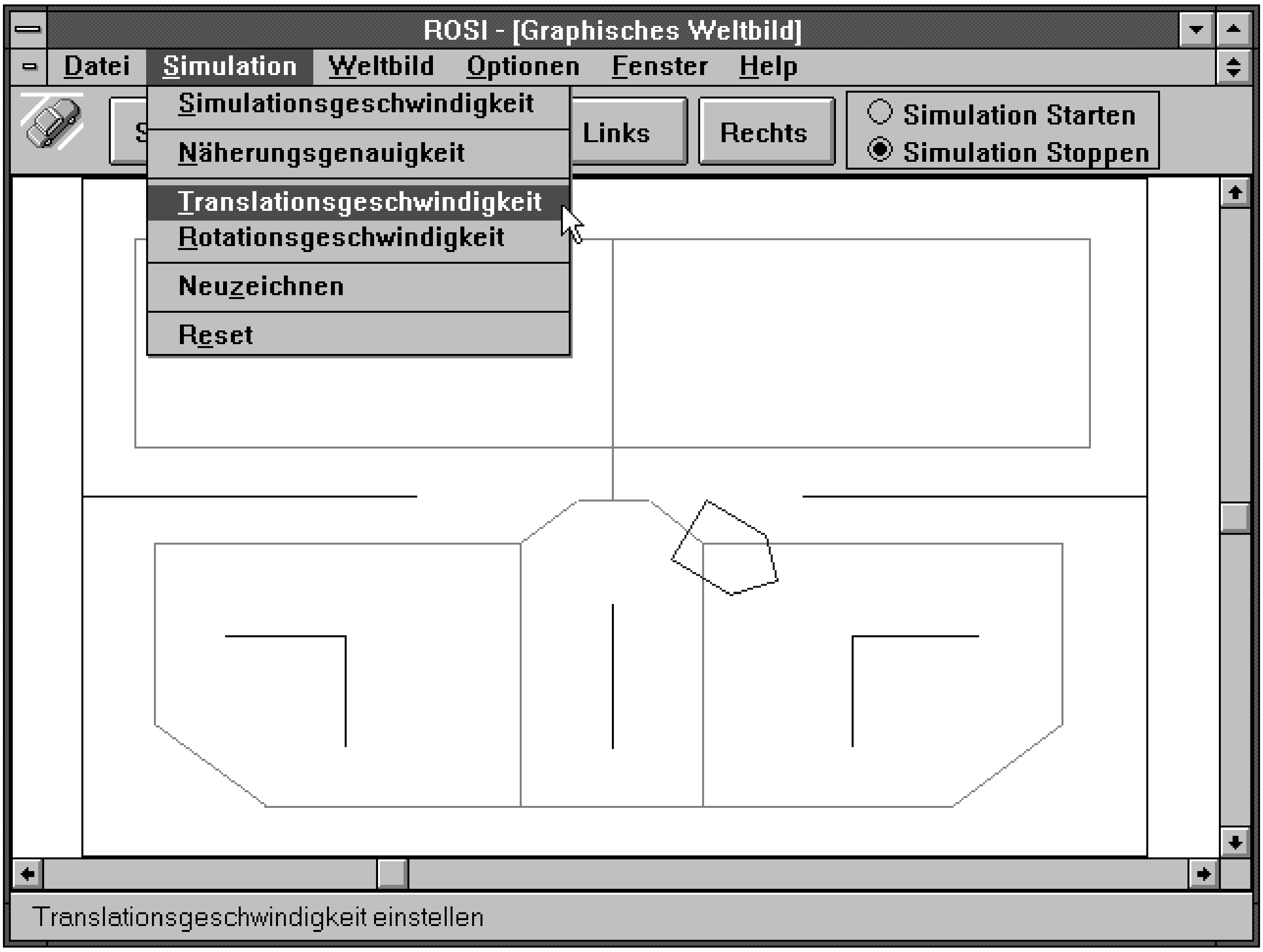


B.7.2 Näherungsgenauigkeit

Die Näherungsgenauigkeit gibt an, ab wann der Roboter aus der Perspektive des Simulators eine Wand berührt.

Je kleiner die Näherungsgenauigkeit ist, desto genauer sind die Simulationsberechnungen. Desto länger dauern aber auch die Simulationsberechnungen, zumindest in der unmittelbaren Umgebung von Wänden.

Die Näherungsgenauigkeit ist auch maßgebend dafür, wann ein Mikroschalter gegen eine Wand anschlägt.

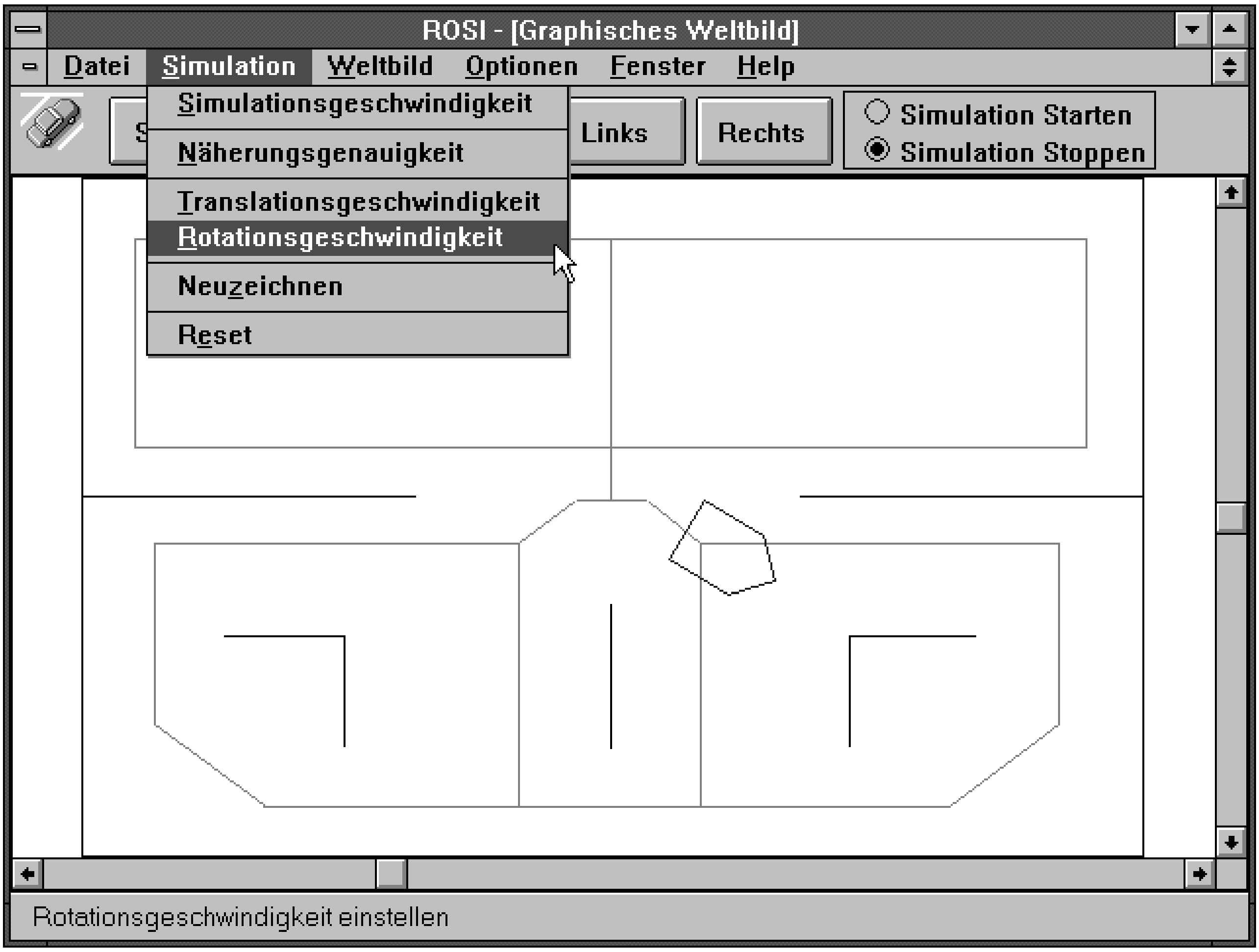


B.7.3 Translationsgeschwindigkeit

Die Translationsgeschwindigkeit gibt die Schrittgröße für die Translationsschritte an.

Je kleiner die Translationsschritte sind, desto genauer sind die Simulationsberechnungen. Desto länger dauern aber auch die Simulationsberechnungen.

Wird die Translationsgeschwindigkeit zu groß gewählt, kann es passieren, daß die Schritte über mehrere Meter gehen. In diesem Fall werden alle Hindernisse übersehen, die zwischen zwei Schritten liegen. Man sollte die Translationsgeschwindigkeit so klein wählen, daß die Schrittweite kleiner als die Roboterlänge ist. Der Simulator überprüft dies nicht, um der Phantasie des Experimentators keine Grenzen zu setzen.

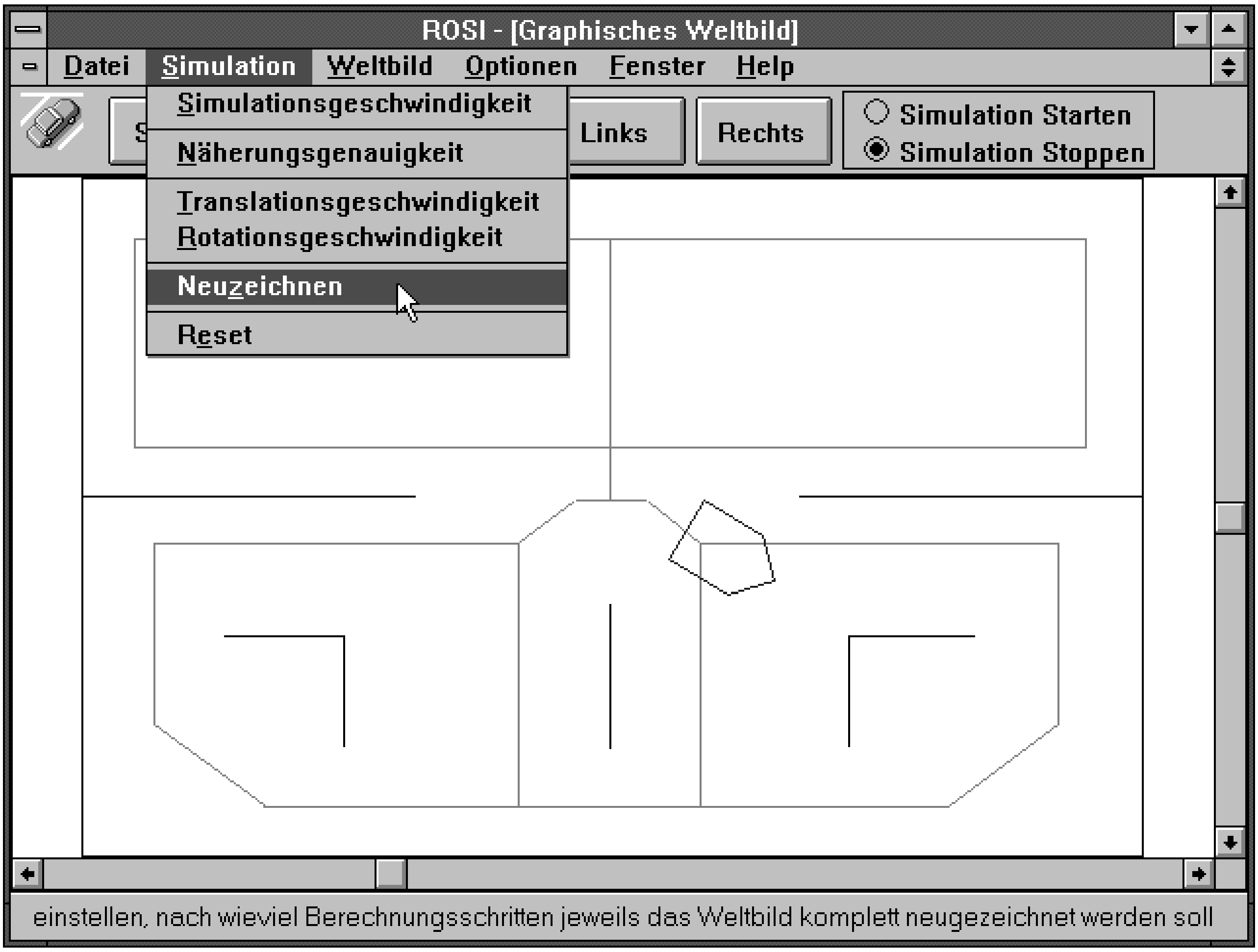


B.7.4 Rotationsgeschwindigkeit

Die Rotationsgeschwindigkeit gibt die Schrittgröße für die Rotationsschritte an.

Je kleiner die Rotationsgeschwindigkeit ist, desto genauer sind die Simulationsberechnungen. Desto länger dauern aber auch die Simulationsberechnungen.

Wenn die Rotationsgeschwindigkeit sehr hoch ist, werden entsprechend große Rotationsschritte gemacht. Dabei könnten eventuell Wandecken vom Simulator übersehen werden, an denen der Roboter normalerweise hängenbleiben würde.



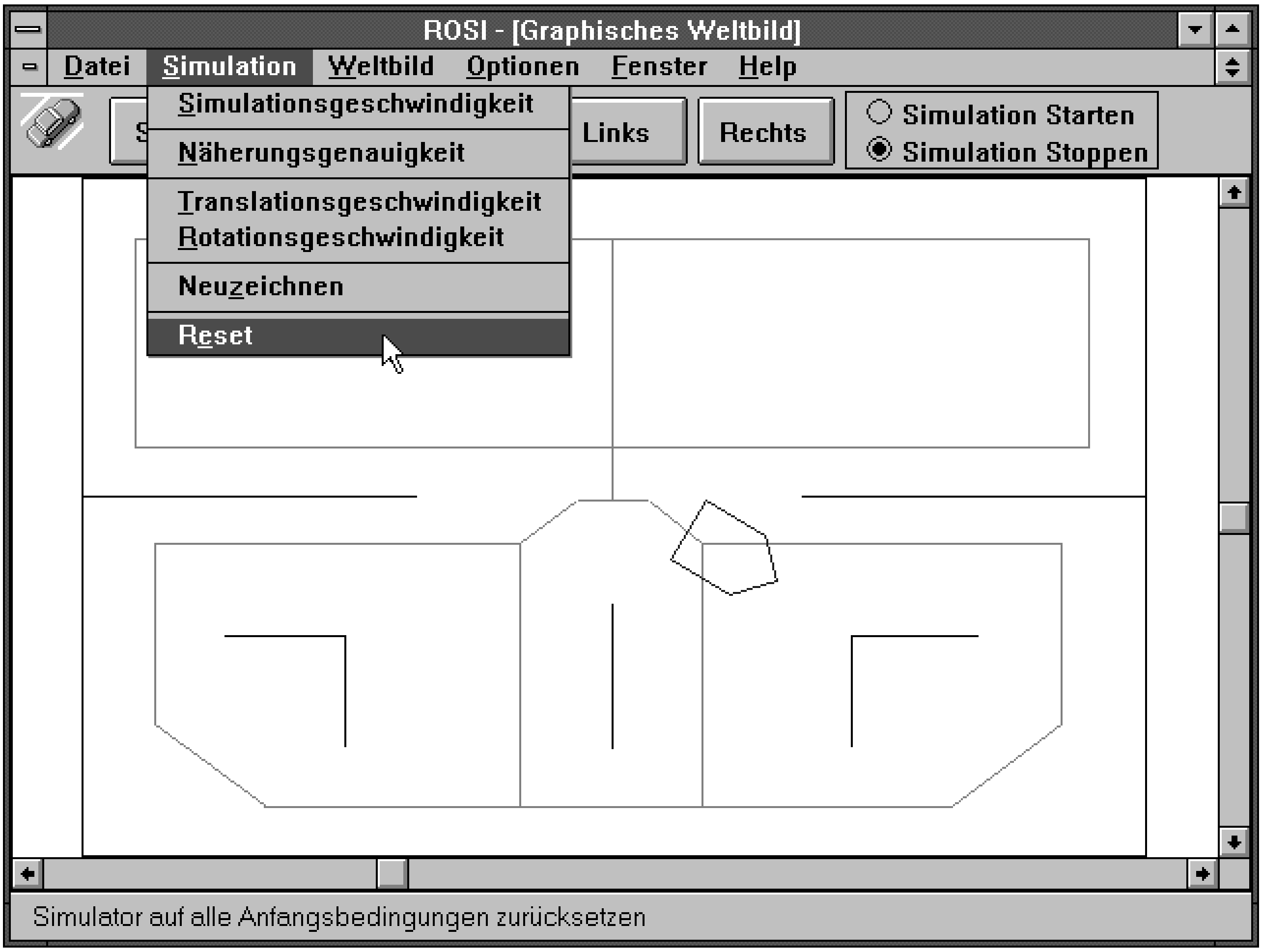
B.7.5 Neuzeichnen

Mit diesem Menüpunkt wird angegeben, nach wievielen Simulationsschritten das Spielfeld und der Roboter jeweils komplett neu gezeichnet werden.

Der kleinstmögliche Wert ist 1. In diesem Fall wird nach jedem Simulationsschritt das gesamte Spielfeld einmal neu gezeichnet.

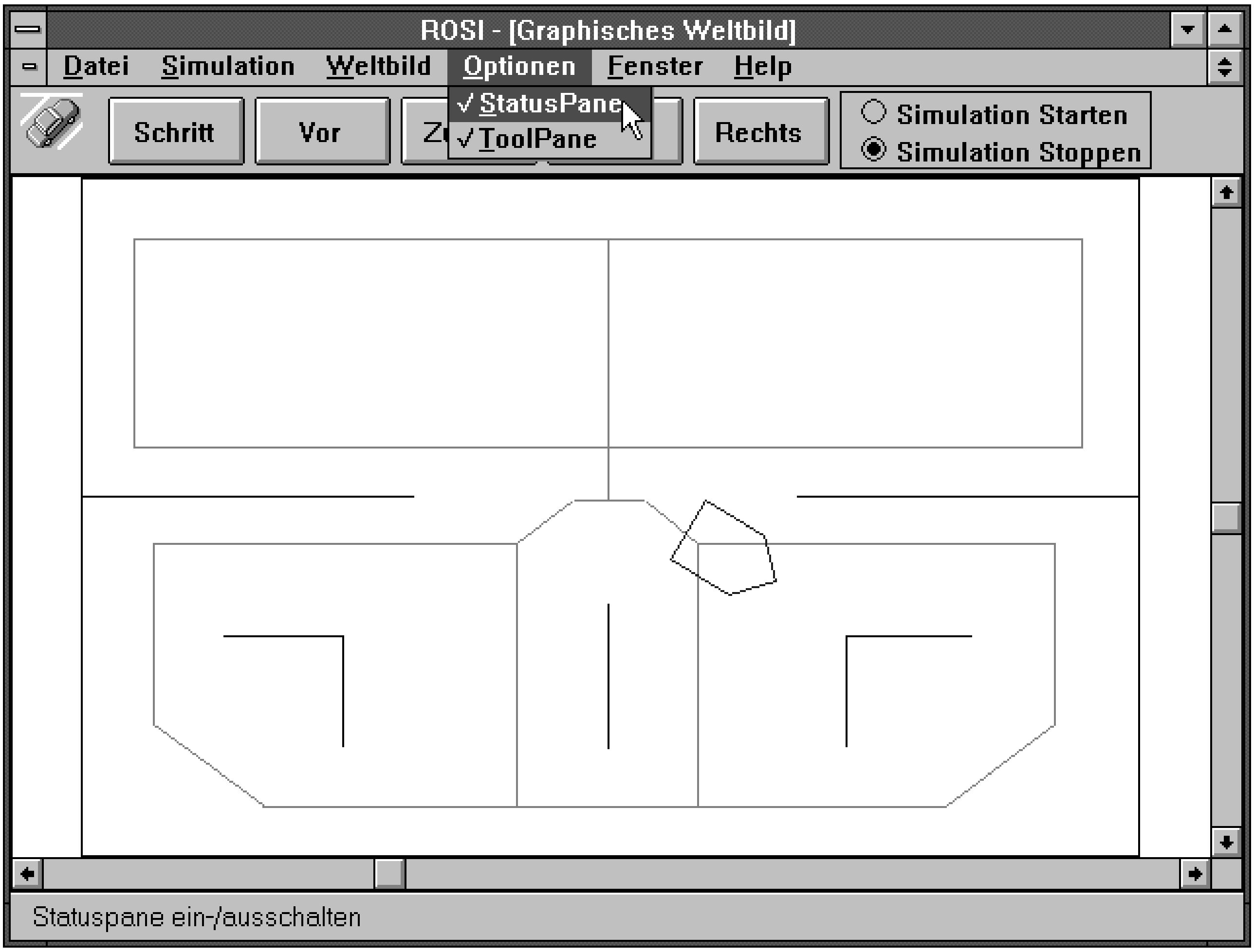
Das Neuzeichnen ist insofern sinnvoll, da der Roboter gelegentlich Orientierungsmarkier-ungen überfährt und zerstört. Außerdem werden Wände gelegentlich etwas *abgeschabt*, wenn der Roboter nahe entlang der Wände fährt. Je gröber die Darstellung eines Spielfeldes ist, desto eher werden die Wände abgeschabt.

Der *Neuzeichnen*-Menüpunkt im *Weltbild*-, *Roboter*- und *Spielfeld*-Menü aktualisiert unmittelbar einmal die gesamte Darstellung, wogegen dieser Menüpunkt für die automatische Darstellungserneuerung ist, während die Simulationsschleife läuft.



B.7.6 Reset

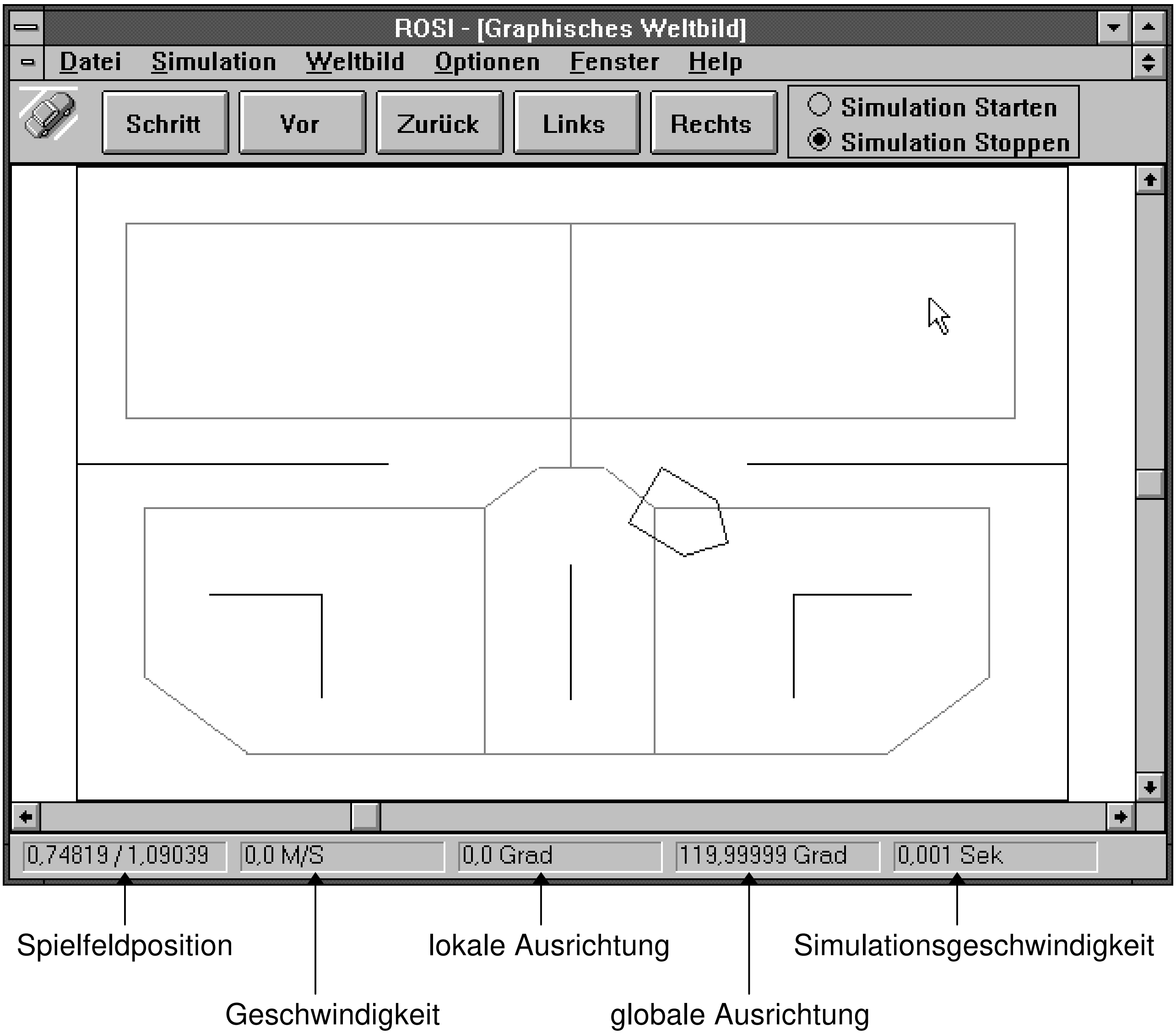
Einige Standardeinstellungen des Simulators werden mit Hilfe dieses Menüpunktes auf ihre ursprünglichen Werte gesetzt.



B.8 Statuszeile anzeigen

Mit Hilfe dieses Menüpunktes wird die Statuszeile entweder angezeigt oder versteckt.

Das Verstecken der Statuszeile ist dann sinnvoll, wenn die Statusinformationen nicht so wichtig sind oder von einem anderen Programm dargestellt werden, während das Spielfeld dadurch etwas größer wird.



B.8.1 Statuszeile

Auf der Statuszeile werden einige nützliche Informationen angezeigt.

Ganz links wird die x-y-Spielfeldposition, auf der sich der Roboter gerade befindet, angezeigt. Diese Position bezieht sich auf den Drehpunkt des Roboters.

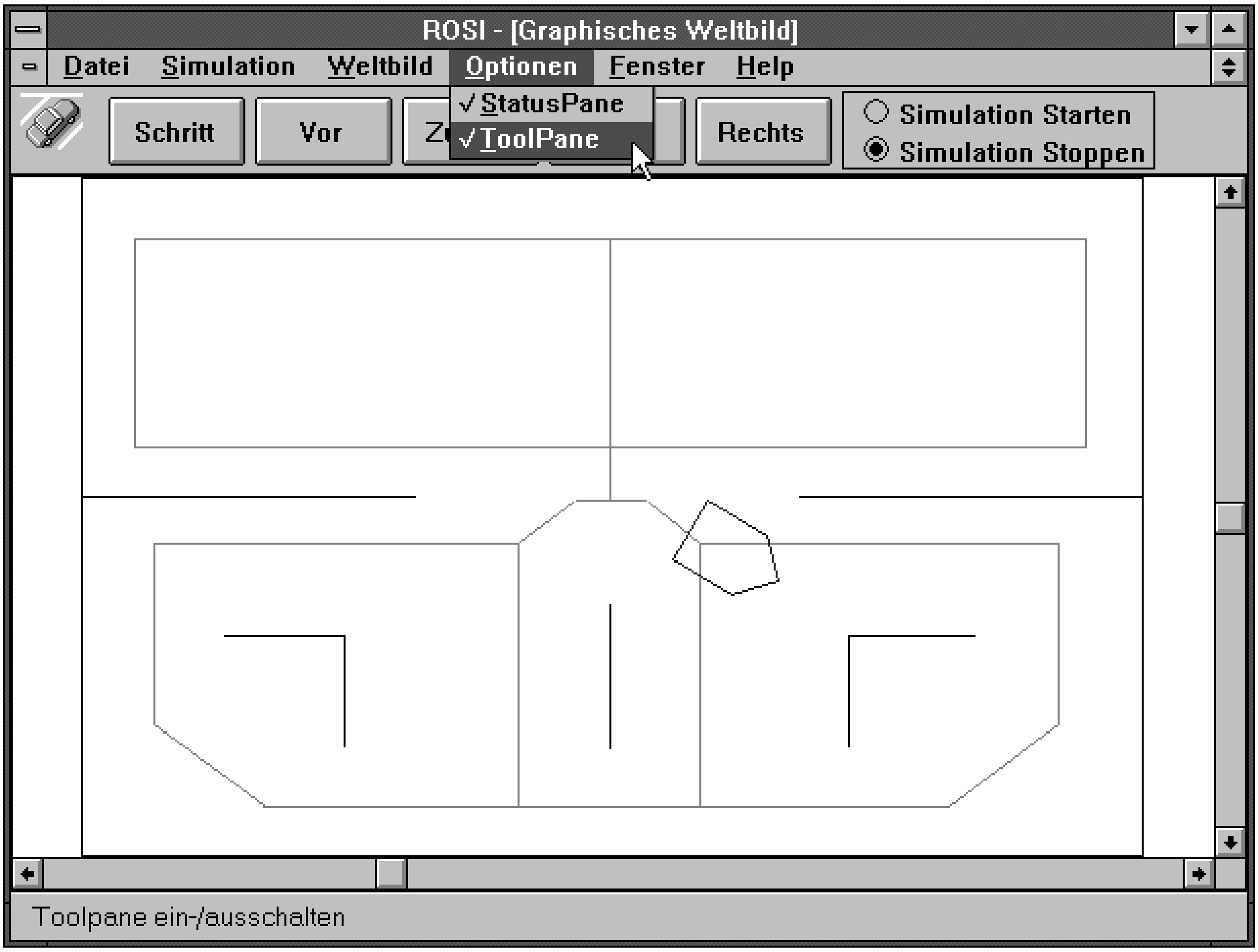
Die Geschwindigkeit [Meter/Sekunde] zeigt die Größe der Translationsschritte.

Die lokale Ausrichtung für den Roboter ist seine Steuerradausrichtung. D.h. je größer dieser Wert ist, desto größer sind die Rotationsschritte.

Die globale Ausrichtung relativ zum Spielfeld zeigt, wo sich die Robotervorderseite befindet. 0° bezieht sich z.B. darauf, daß die Robotervorderseite nach oben zeigt. Bei 180° zeigt die Robotervorderseite nach unten.
Entsprechend 119,99° zeigt der Roboter der Abbildung nach rechts unten.

Das Aufblinken der Translationsgeschwindigkeit zeigt wie schnell die Simulation läuft. Die Anzeige leuchtet abwechselnd gelb und rot auf.

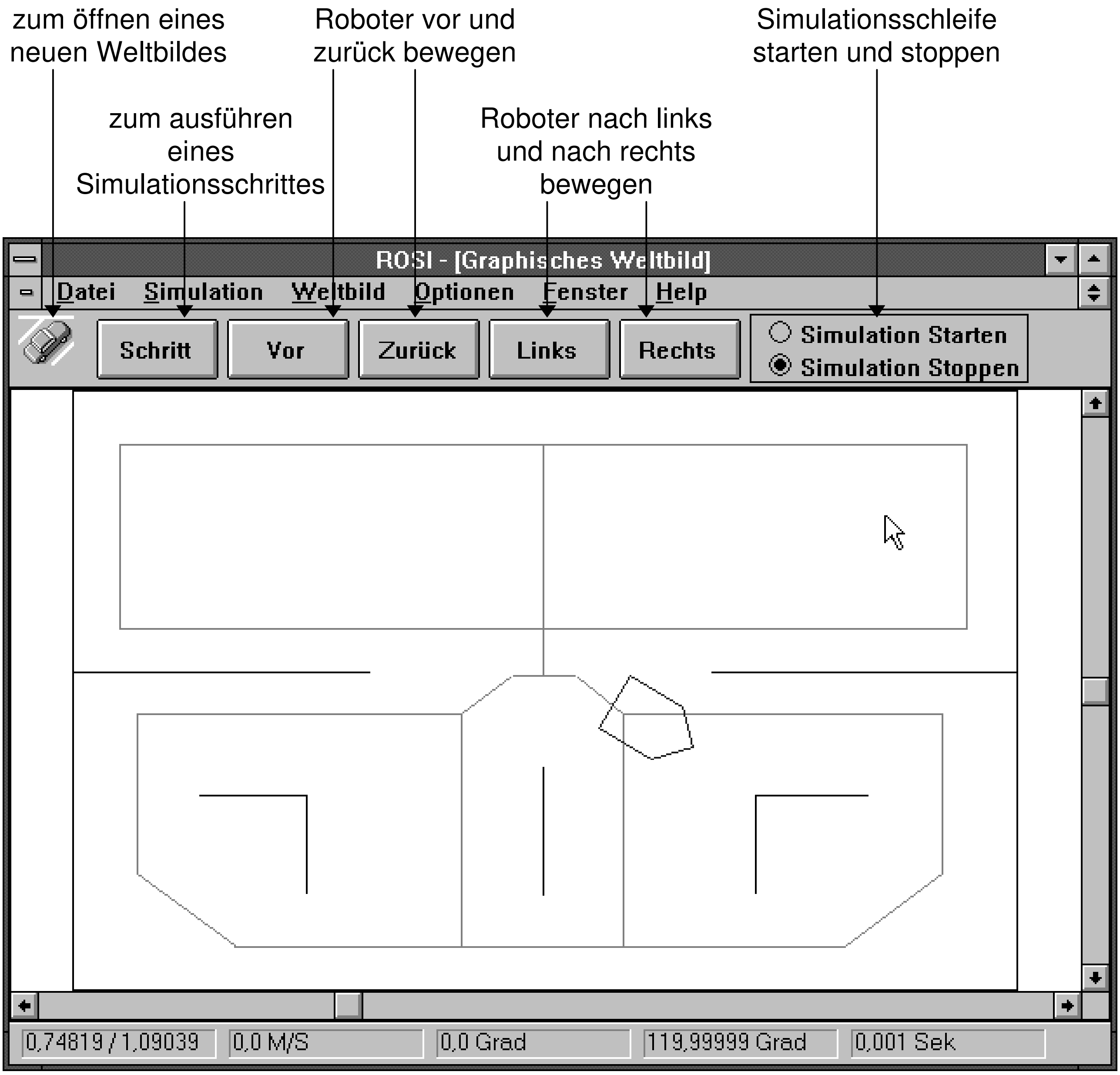
Blinkt diese Anzeige nicht obwohl der Schalter Start Simulation aktiviert ist, dann ist entweder die Simulationsgeschwindigkeit zu langsam eingestellt, Menü Simulation / Simulationsgeschwindigkeit, oder die Berechnungen dauern tatsächlich sehr lange oder der Simulator hängt in einer Totschleife.



B.9 Toolpane anzeigen

Die Toolpane kann mit diesem Menüpunkt angezeigt oder versteckt werden.

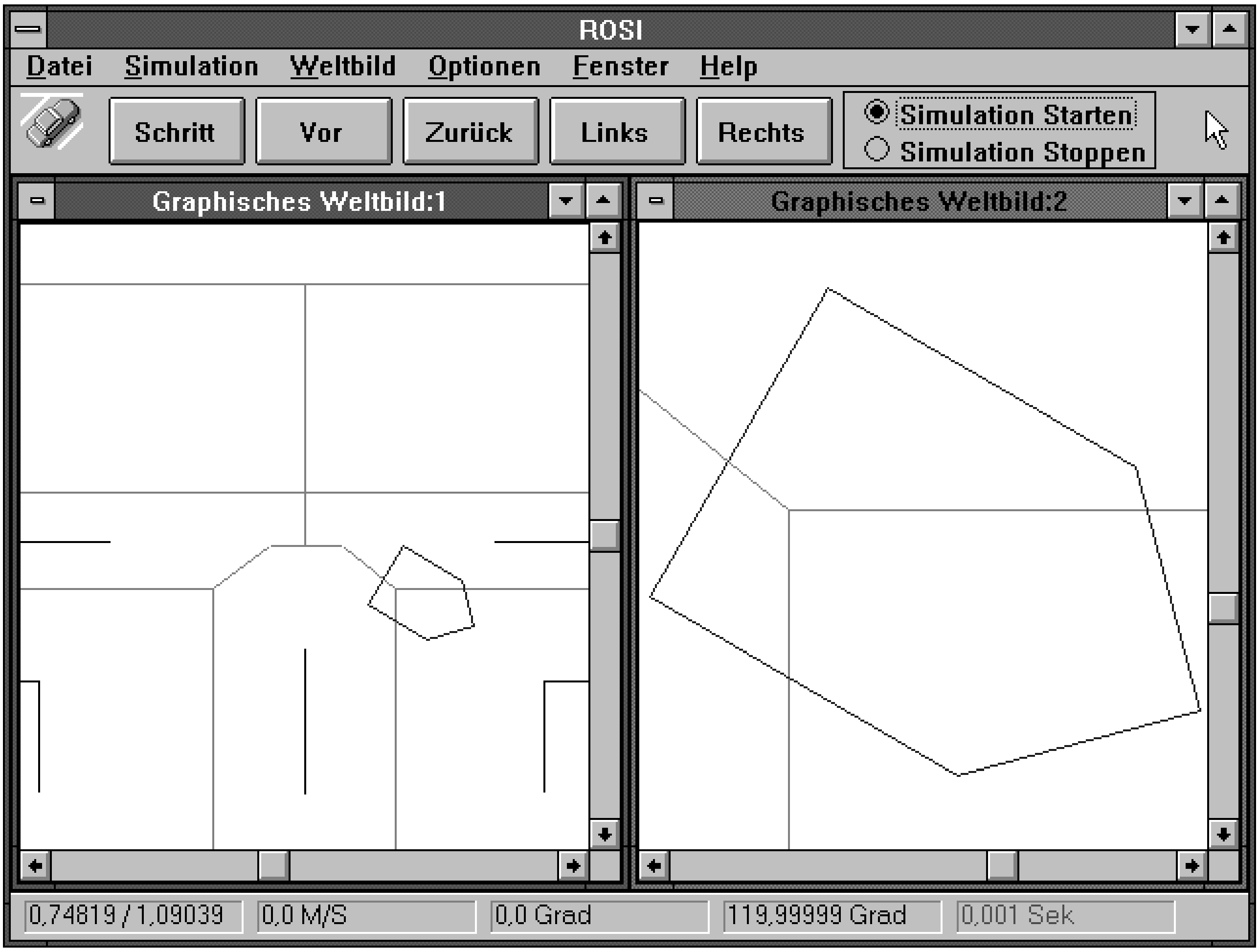
Wenn man die Simulationsschleife nicht benutzt und statt dessen ein Steuerungsprogramm den Roboter steuert, wird die Toolpane nicht gebraucht. Wenn sie versteckt ist, dann ist das Spielfeld auch etwas größer.



B.9.1 Toolpane

Mit Hilfe des Schalters Start Simulation auf der Toolpane wird die Simulationsschleife ROSIs gestartet. D.h. es wird eine Schleife aufgerufen, in der pro Durchlauf die Position des Roboters neu berechnet wird. Hierzu werden die Einstellungen zur Rotation, Links und Rechts in der Toolpane, sowie die Translationseinstellungen, Vor und Zurück, genommen. Für jeden Durchlauf leuchtet unten rechts auf der Statuszeile gelb bzw. rot die Translationsgeschwindigkeit auf.

Mittels Step wird bloß ein einziger Simulationsschritt ausgeführt. Step macht nur Sinn, wenn der Schalter auf Stop Simulation steht.



B.10 Zoomen des Spielfeldes

Wenn die Simulationsschleife ROSIs deaktiviert ist, der Schalter in der Toolpane also auf Simulation Stoppen steht, dann kann das Spielfeld herangezoomt werden.

Dazu drückt man die linke Maustaste und zeichnet von links oben nach rechts unten ein Rechteck, das den zu zoomenden Ausschnitt ungefähr beschreibt. Ab einer gewissen Genauigkeit kann es beim weiteren heranzoomen zu fehlerhaften Darstellungen und Verhalten des Simulator führen. *Runtime error* werden aber abgefangen und angezeigt. Nach jedem *Runtime error* kann gewöhnlicherweise normal weitergearbeitet werden. Im Notfall schließt man alle Fenster, bestätigt alle *Runtime error* und arbeitet weiter indem man neue Weltbilder öffnet. Sollte das auch nicht helfen, muß die Anwendung beendet und neu gestartet werden. Auf diese Weise werden alle Variablen wieder mit angemessenen Werten initialisiert.