Dokumentation zur Aufgabe 2 – Tankomatik

Programmdokumentation von

Christian Siewert

für den 27. Bundeswettbewerb Informatik

1 Aufgabenstellung

Die zweite Aufgabe des 27. BWINF besteht in der Simulation von Ölpreisen an einem Spotmarkt (z. B. In Rotterdam) und darauf basierend Preisentwicklungen und Kundenverhalten an Tankstellen.
Dabei sollen diese Preisentwicklungen bzw. das Kundenverhalten an genau zwei Tankstellen simuliert werden. Beide Tankstellen arbeiten nach unterschiedlichen Methoden zur Preisfestlegung. Die Tankstelle Asso differenziert dabei die Anzahl der Kunden der letzten Stunde. Die Tankstelle Scholl setzt den Preis immer auf fünf Cent über den aktuellen Einkaufspreis.
Darüber hinaus soll untersucht werden welche der beiden Tankstellen langfristig gesehen den höheren Gewinn macht und wie Asso seine Preispolitik optimieren könnte.
Es ist bekannt an welcher Tankstelle Autos tanken und wie oft pro Sekunde ein Auto an der jeweiligen Tankstelle einbiegt.

2 Lösungsidee

Das schwierige an dieser Aufgabe ist die Simulation von realistischen Preisentwicklungen am Spotmarkt. Kein Rohstoffpreis schwankt so stark wie Öl. Dies beruht auf der Tatsache das der Ölpreis von vielen verschiedenen Faktoren abhängig ist. Diese Faktoren (z. B. erhöhte Nachfrage aus Schwellenländern, politische Krisen, etc.) müssen bei der Simulation des Ölpreises mit einbezogen werden. Um ein gewisses Maß an Realismus zu erzeugen sollten möglichst viele Einflüsse in der Simulation implementiert werden. Diese Ereignisse können durch bestimmte Zufallszahlen ausgelöst werden. Jeder dieser Einflüsse verändert den Ölpreis welcher dann sofort als aktueller Einkaufspreis festgelegt wird. Den Anfangswert für den Ölpreis kann man problemlos im Internet recherchieren. Ausgehend von dieser Simulation können dann Ölpreise an den Tankstellen festgelegt werden. Dabei ist es wichtig das beide Tankstellen am Anfang den Ölpreis am Spotmarkt abfragen und speichern.

Um etwas mehr Realismus zu erzeugen sollten beide Tankstellen noch ein Fixum auf diesen Ölpreis aufschlagen. Dieses Fixum sollte sich aus Mineralölsteuer, Mehrwertsteuer, einer spezifischen Tankstellenmarge (welche bei beiden gleich sein muss) und einem Beitrag für den Erdölbevorratungsverband zusammensetzen. Nachdem beide Tankstellen zum ersten Mal ihre Preise festgelegt haben, kann die eigentliche Simulation gestartet werden. Dabei sind die Regeln aus der Aufgabenstellung unbedingt zu beachten. Die Autos fahren nicht immer an den Tankstellen vorbei. Es ist bekannt das jede Sekunde ein Auto vorbeifährt und mit einer Wahrscheinlichkeit von 1/60 tankt. Dies kann man mit Zufallszahlen lösen. Wenn man pro Sekunde eine Zufallszahl zwischen 1 und 60 ermittelt und diese einen bestimmten Wert hat (z. B. den Wert 1), dann tankt dieses Auto. Darüber hinaus muss ein System implementiert werden welches den Autos mitteilt an welcher Tankstelle sie tanken sollen. Dies kann man ebenfalls mit Zufallszahlen realisieren. Wenn man eine Zufallszahl zwischen 1 und 10 bestimmt und diese im Bereich 1 bis 6 liegt, so tankt das jeweilige Auto an der preisgünstigeren Tankstelle. Bei 7 bis 8 tankt das Auto bei Asso und bei 9 bis 10 bei Scholl. Um die Anzahl der Kunden an der jeweiligen Tankstelle zu bestimmen könnte man einen tankstellenspezifischen Zähler implementieren, der immer inkrementiert wird sobald ein Auto tankt. Durch diesen Zähler ist auch die Differenzierung der Anzahl der Kunden durch Asso möglich. Dadurch ist Asso dann in der Lage mit Preisveränderungen auf die Anzahl der Kunden der letzte Stunde zu reagieren. Natürlich muss dieser Zähler dann jede Stunde auf Null gesetzt werden.
Scholl setzt den Preis immer auf 5 Cent über den aktuellen Einkaufspreis. Deshalb fragt Scholl alle 6 Stunden den aktuellen Ölpreis am Spotmarkt Öl ab. Dies tut allerdings auch Asso. Dabei speichert Asso diesen neuen Ölpreis nur für die eigene Differenzierung bzgl. der Neufestlegung der Preise. Sollte also Asso seinen Preis verändern und sollte dieser neue Preis unterhalb dem aktuellen Einkaufspreis liegen, so wird der Einkaufspreis verlangt. Dies kann man realisieren indem Asso alle 6 Stunden den aktuellen Ölpreis abfragt und diesen in einer Art Puffer speichert. Sollte nun eine Preisveränderung bei Asso stattfinden, so sollte Asso erst durch Bedingungen prüfen ob dieser neue Preis unterhalb dem Einkaufspreis liegt.
Der Gewinn der beiden Tankstellen soll sich aus Anzahl_Kunden * akt_Preis ergeben.
Das wichtigste Element in dieser Simulation ist eine Komponente die in

periodischen Abstände Befehle ausführt. Dies könnte man mit einem Timer realisieren. Das Standardinterval dieses Timers sollte auf eine Sekunde gesetzt werden.
Dadurch können dann pro Timerinterval (pro Sekunde) Ereignisse ausgelöst werden, welche den Ölpreis beeinflussen. Darüber hinaus kann pro Sekunde geprüft werden ob ein Auto tankt. Darüber hinaus sollte ein Zähler mitlaufen welcher pro Timerinterval differenziert wird. Sollte genau eine Stunde vergangen sein, so sollen die neuen Preise festgelegt werden. Außerdem ist dieser Zähler auch wichtig alle 6 Stunden den aktuellen Ölpreis am Spotmarkt abzufragen. Eine Übersicht über alle Teilprobleme und deren Lösungsidee finden Sie in Anlage A.

3 Programm-Dokumentation

Als erstes möchte ich das Design meiner Lösung vorstellen. Mein Programm besteht aus einem Hauptfenster, mehreren Labels für die Ausgaben, vier Buttons um Aktionen auszuführen und einem Schieberegler um das Timerinterval zu ändern.
Die Implementierung des Timers ließ sich leicht realisieren. GTK+ unterstützt standardmäßig Funktionen zum Hinzufügen oder Entfernen eines Timers. Nach dem Druck auf den Button zum Starten der Simulation wird eine Funktion aufgerufen die den Timer startet. Diese Funktion bekommt als Parameter u. a. das Timerinterval übergeben. Diesen Parameter habe ich mit dem Wert 1000 initialisiert. Dadurch wird dann alle 1000 Millisekunden die Timerfunktion ausgeführt in der praktisch der Kern der Anwendung arbeitet. Der Schieberegler lässt genau 4 Zustände zu (Echtzeit, Schnell, Sehr schnell, Ultraschnell). Je nachdem wie man diesen Schieberegler bedient wird der alte Timer entfernt, ein neues Interval festgelegt (1000, 100, 5, 1) und der neue Timer gestartet. Dadurch ist es möglich das Programm in sehr schneller Zeit ablaufen zu lassen. Nachdem der Timer gestartet wurde, hat der Benutzer die Möglichkeit die Anwendung zu pausieren. Beim Druck auf den Button „Pause“ wird der Timer entfernt und die Simulation somit unterbrochen. Die Simulation wird fortgesetzt wenn man auf den Button „Anwenden“ drückt. Damit der User immer die genaue Uhrzeit der Anwendung nachvollziehen kann, wurde ein Label implementiert welches die aktuelle Uhrzeit anzeigt.

Dies übernimmt die Funktion `show_time(Tankomat *program)`. Beim Start der Simulation wird der Ölpreis ausgegeben. Dabei verwende ich zwei Labels welche den Ölpreis in „Euro pro Barrel“ und „Cent pro Liter“ anzeigen. Der Umrechnungsfaktor beträgt hierbei 158,98. Zum Ausgeben des aktuellen Ölpreises wird die Funtkion `ov_event_show_oelpreis (Tankomat *program)` benutzt. Anschließend können Asso und Scholl zum ersten Mal diesen Preis abfragen und ihren eigenen festlegen. Sie tun dies indem sie einfach die Ölpreis (Cent pro Liter – Wert) kopieren und ein Fixum aufschlagen. Scholl erhöht diesen Preis noch um weitere 5 Cent. Das Fixum setzt sich aus Mineralölsteuer, Mehrwertsteuer, einem Beitrag für den EBV und einer Marge (Transportkosten, Löhne, etc.) zusammen und beträgt 10 Cent. Anschließend werden diese Preise ausgegeben und die eigentliche Simulation kann nun starten.
Der eigentliche Simulationsablauf wird durch eine periodische Inkrementierung der Zeit realisiert. Dadurch ist es auch möglich auf volle Stunden zu reagieren aber dazu später mehr. In jedem Timerinterval wird zufallsbasierend ein Ereignis ausgelöst welches den Ölpreis beeinflusst. Dieses habe ich, wie in der Lösungsidee beschrieben, mit Zufallszahlen realisiert. Dazu wird innerhalb der Timerfunktion eine Zufallszahl zwischen 1 und 100 ermittelt. Sollte diese den Wert 36 besitzen (Wahrscheinlichkeit von 1%) so wird eine weitere Zufallszahl zwischen 1 und 1000 bestimmt. Diese Zahl wird dann differenziert und je nach Wert wird ein Ereignis ausgelöst. Ereignisse sind in dem Fall nichts anderes als Funktionen denen bestimmte Parameter mit übergeben werden. Durch diese Ereignisse steigt oder fällt der Ölpreis. Der jeweilige Wert wird auch zufällig bestimmt um die ganze Sache etwas dynamischer zu gestalten. Sie finden eine Übersicht über alle Zahlen, deren Ereignisse und die zugehörigen Veränderungen des Ölpreises in Anlage B. Natürlich müssen ein paar Besonderheiten beachtet werden. So kann sich zum Beispiel eine Irak-Krise nicht verschärfen wenn es im Irak noch gar keine Krise gibt. Dies wurde durch Parameter bewerkstelligt, welche den aufzurufenden Funktionen übergeben werden. Im folgenden möchte ich diese Funktionen erläutern:

void on_event_krisen (gchar *gebiet, gboolean krise_aktiv, gboolean steigen)

Diese Funktion ist dafür zuständig eine Krise in einem bestimmten Gebiet auszulösen. Das Gebiet wird als Char Array übergeben. Außerdem wird eine boolesche Variable übergeben ob angibt ob die Krise im jeweiligen Gebiet schon aktiv ist. Somit kann zwischen Entstehung und Verschärfung der Krise unterschieden werden. Der dritte Parameter ist wieder vom Typ Boolean. Hier wird angegeben ob der Ölpreis steigt oder fällt. Somit kann zusätzlich zu Entstehung oder Verschärfung auch zwischen Entspannung unterschieden werden. Ein Beispiel könnte wie folgt aussehen:

```
on_event_krisen("Irak", krise_irak, TRUE);
```

Hier wird eine Krise im Irak ausgelöst. Dabei wird der Parameter „krise_irak“ übergeben welcher angibt ob die Krise schon aktiv ist. Wenn zum ersten Mal diese Funktion aufgerufen wird hat diese Variable den Wert „false“. Somit entsteht hier also eine Krise im Irak. Aufgrund dessen das der dritte Parameter (steigen) auf „true“ steht wird der Ölpreis steigen. Ein anderes Beispiel:

```
on_event_krisen("Irak", krise_irak, FALSE);
```

Hier sieht man das der dritte Parameter auf „false“ steht. Somit handelt es sich um eine Entspannung der Irak-Krise. Dies wird natürlich nur dann realisiert wenn vorher eine Krise im Irak existierte.

void on_event_kriege (gchar *gebiet, gboolean krieg_aktiv, gboolean steigen)

Die Funktion zum Auslösen der Kriege im Iran oder im Irak funktioniert genauso wie die Funktion zum Auslösen von Krisen. Allerdings fließen hier, wie oben in der Tabelle zu sehen, andere Geldsummen.

void on_event_nachfrage (gchar *gebiet, gboolean steigen)

Diese Funktion löst erhöhte oder verminderte Nachfrage eines bestimmtes Landes aus.

In diesem Fall habe ich mich für Schwellenländer wie China, Indien und Brasilien entschieden. Auch hier wird wieder eine boolesche Variable übergeben die angibt ob der Ölpreis fällt oder steigt. Somit kann zwischen einer erhöhten oder verminderten Nachfrage unterschieden werden. Es folgt ein Beispiel:

```
on_event_nachfrage("Indien", TRUE)
```

Man sieht das hier eine erhöhte Nachfrage aus Indien ausgelöst wird was folglich den Ölpreis in die Höhe treibt.

`on_event_opec(gboolean steigen)`

Internationale Organisationen wie die OPEC haben einen großen Einfluss auf den Ölpreis. In diesem Fall habe ich mich für Ölpreisprognostizierungen entschieden, welche die OPEC veröffentlicht. Je nachdem ob die OPEC nun sinkende oder steigende Ölpreise prognostiziert, steigt oder fällt der Ölpreis.

`on_event_wechselkurs(gboolean steigen)`

Einer weitere Einflussgröße auf den Ölpreis ist der Dollarkurs. Je nachdem ob dieser steigt oder fällt, steigt oder fällt auch der Ölpreis.

Innerhalb jeder dieser Funktionen wird ein zufallsbasierender Geldbetrag ermittelt und anschließend auf den aktuellen Ölpreis aufgeschlagen oder abgezogen.
Außerdem wird das jeweilige Ereignis samt Tag und Uhrzeit in dem er auftrat in einem Textfeld ausgegeben. Dazu wird die Funktion `on_event_show_text (gchar *text)` welche als Parameter den auszugebenden Text übergeben bekommt aufgerufen. Somit kann ein User zu jedem Zeitpunkt sehen, wann welche Ereignisse ausgelöst wurden und wie sich selbige auf den Ölpreis niederschlugen.

Somit ist ein (einigermaßen) realistisches System zur Simulation des Ölpreises sichergestellt. In der Anwendung kam es teilweise zu starken Schwankungen des Ölpreises. Diese Schwankungen reichten von unter 50 Euro pro Barrel bis zu 120 Euro pro Barrel. Selbst diese 120 Euro pro Barrel sind seit Sommer diesen Jahres nicht mehr utopisch.

Ausgehend von der Simulation des Ölpreises wurde als nächstes die Preisentwicklungen an Tankstellen und deren Kundenverhalten simuliert. In der Aufgabe wurde gesagt, dass pro Sekunde ein Auto an einer Tankstelle vorbeifährt und mit einer Wahrscheinlichkeit von 1 / 60 tankt. Aus diesem Grund wurde eine Zufallszahl zwischen 1 und 60 ermittelt. Beträgt diese 30 (festgelegter Wert), so tankt der jeweilige Kunde. Anschließend wird ermittelt an welcher Tankstelle der Kunde nun tankt. 20% der Kunden gehen zu Asso, 20% zu Scholl und 60% tanken bei der günstigeren Tankstelle. Im Programm wird also wieder eine Zufallszahl bestimmt die zwischen 1 und 10 liegt. Besitzt diese Zahl den Wert 7 oder 8 so tankt der Kunde bei Asso. Beträgt sie hingegen 9 oder 10 so tankt er bei Scholl. Liegt die Zahl zwischen 1 und 6 so wird geprüft welche der beiden Tankstellen die günstigere ist. Der Kunde tankt anschließend bei der günstigeren Tankstelle. Sollte keine der beiden Tankstellen günstiger sein so fährt der Kunde weiter und tankt bei Tatol.
Je nachdem wo der Kunde tankt werden die Anzahl der Kunden der Tankstelle erhöht, diese Anzahl ausgegeben und der Gewinn der Tankstelle erneut kalkuliert. Der Gewinn ergibt sich aus Anzahl_Kunden * Aktueller_Preis. Für Asso wird noch die Variable „kunden_asso_std“ inkrementiert. Dieser Wert wird zu jeder vollen Stunde auf null gesetzt. Die Funktionen die die Anzahl der Kunden an der jeweiligen Tankstelle ausgeben heißen `akt_kunden_asso(Tankomat *program)` und `akt_kunden_scholl(Tankomat *program)`. Beide Funktionen sind vom Typ `void`. Die Funktionen zum Berechnen des Gewinns an der jeweiligen Tankstelle heißen `calc_gewinn_asso(Tankomat *program)` und `calc_gewinn_scholl(Tankomat *program)`. Sie sind ebenfalls vom Typ `void`.

Nachdem eine Stunde im Programm vergangen ist kalkulieren kalkuliert Asso erneut ihre seine Preis. Sie tut dies mit der Funktion `calc_preis_asso(Tankomat *program)`.

Innerhalb dieser Funktion werden die Anzahl der Kunden der letzten Stunde differenziert. Bei mehr als 35 Kunden erhöht Asso den Preis um einen Cent. Bei weniger als 25 Kunden verringert sie den Preis um einen Cent. Sollte dieser Preis unterhalb des aktuellen Einkaufspreises liegen, so entspricht der neue Preis dem aktuellen Einkaufspreis. Asso speichert darüber hinaus alle 6 Stunden den aktuellen Einkaufspreis in einer Puffervariable. Dieser Preis wird dann ausgegeben und die Anzahl der Kunden der letzten Stunde wird auf null gesetzt.
Die Preispolitik von Scholl sieht vor alle 6 Stunden den Einkaufspreis am Spotmarkt Öl abzufragen und 5 Cent zu addieren. Dies wird durch die Funktion `calc_preis_scholl(Tankomat *program)` realisiert.
Als kleines Extra habe ich noch Funktionen geschrieben, welche die Simulation pausieren und zurücksetzen. Dies wird durch die Funktion `on_cmd_pause_clicked` respektive `on_cmd_reset_clicked(GtkObject *object, Tankomat *program)` ermöglicht.
Die Funktion zum Pausieren der Simulation entfernt den Timer und die Funktion zum Zurücksetzen der Simulation setzt bestimmte Variablen auf ihren Anfangswert zurück und entfernt ebenfalls den Timer.
Somit sind alle geforderten Aufgaben erfüllt und die Simulation kann durchgeführt werden. Eine Eingabe durch einen Benutzer ist nicht nötig respektive möglich. Er kann allerdings die Geschwindkeit des Programms beeinflussen. Darüber hinaus kann er die Simulation pausieren, fortsetzen und zurücksetzen.
Eine vollständige Übersicht aller Funktionen finden Sie in Anlage C. Darüber hinaus finden Sie in Anlage D eine Übersicht der wichtigsten Variablen und deren Funtkion. Außerdem habe ich noch ein Nassi-Shneiderman-Diagramm erstellt welches den Kern des Programms visualisiert. Sie finden dieses Struktogramm in Anlage E. Beachten Sie bitte das dieses Diagramm als idealisiert zu betrachten ist und aufgrund der Komplexität der Aufgabenstellung nicht hunderprozentig mit dem Quellcode übereinstimmt.

4 Programm-Ablaufprotokoll

Im folgenden wird ein Beispiel des Programmablaufes gezeigt. Dabei werde ich mit Tabellen arbeiten und die Simulation einen Tag laufen lassen. Der Anfangszustand sei wie folgt definiert:

Uhrzeit	Ölpreis (Cent / Liter)	Preis Asso	Preis Scholl	Kunden Asso (std)	Ereignis
00:00:00	42,68	1,39	1,44	0	

Wie man sieht fragen am Anfang beide Tankstellen den Ölpreis ab, schlagen verschieden Steuersätze und eine Marge drauf und bieten ihr Benzin dann zum Verkauf an. Scholl ist um 5 Cent teurer, da Scholl immer 5 Cent auf den Einkaufspreis draufschlägt.

Uhrzeit	Ölpreis (Cent / Liter)	Preis Asso	Preis Scholl	Kunden Asso (std)	Ereignis
00:59:20	42,68	1,39	1,44	49	

Aufgrunddessen das Asso billiger ist, tanken statistisch gesehen 80% der Kunden bei Asso. Aus diesem Grund ist Asso bei der Anzahl der Kunden Scholl weit vorraus. Außerdem hat Asso in der letzten Stunde mehr als 35 Kunden gehabt. Deswegen sollte der Benzinpreis bei Asso nun um einen Cent steigen.

Uhrzeit	Ölpreis (Cent / Liter)	Preis Asso	Preis Scholl	Kunden Asso (std)	Ereignis
01:00:01	42,68	1,40	1,44	0	

Asso erhöht tatsächlich seinen Benizinpreis um einen Cent. Asso ist allerdings immer noch billiger als Scholl. Aus diesem Grund wird Asso in den nächsten Stunden den mehr Kunden anziehen.

Uhrzeit	Ölpreis (Cent / Liter)	Preis Asso	Preis Scholl	Kunden Asso (std)	Ereignis
01:45:41	42,67	1,40	1,44	36	Geringere Nachfrage aus Indien
01:59:47	42,67	1,40	1,44	45	
02:01:32	42,67	1,41	1,44	2	
02:55:41	42,67	1,41	1,44	44	Transportkosten steigen
02:59:40	42,67	1,41	1,44	47	
03:01:13	42,67	1,42	1,44	2	
03:06:35	42,67	1,42	1,44	17	Transportkosten sinken
03:59:37	42,67	1,42	1,44	48	
04:00:51	42,67	1,43	1,44	0	
04:59:50	42,67	1,43	1,44	57	
05:00:37	42,67	1,44	1,44	0	

80% der Kunden tankten in den ersten 5 Stunden bei Asso weil Asso immer billiger war als Scholl. Allerdings hob Asso seinen Preis zu jeder vollen Stunde um einen Cent an. Ab 05:00:00 Uihr ist Asso genauso teuer wie Scholl. Aufgrunddessen werden statistisch gesehen nur noch 20% der Kunden bei Asso tanken.

Uhrzeit	Ölpreis (Cent / Liter)	Preis Asso	Preis Scholl	Kunden Asso (std)	Ereignis
05:59:23	42,67	1,44	1,44	8	

Wie man sieht tanken jetzt deutlich weniger Kunden bei Asso. Nun ist es kurz vor 06:00:00. Um 06:00:00 sollte Asso seinen Preis um 1 Cent senken da er weniger als 25 Kunden hatte. Sollte dieser Preis unter dem aktuellen Einkaufspreis liegen, so wird stattdessen dieser verlangt. Zu diesem Zweck wird Asso den aktuellen Einkaufspreis am Spotmarkt abfragen und ihn in einer Puffervariable speichern. Das gleiche gilt auch für Scholl. Scholl wird den Einkaufspreis abfragen und wieder 5 Cent draufschlagen.

Uhrzeit	Ölpreis (Cent / Liter)	Preis Asso	Preis Scholl	Kunden Asso (std)	Ereignis
06:01:39	42,67	1,43	1,44	1	
06:05:02	42,64	1,43	1,44	8	Sinkende Preise durch Preiseprognose
06:59:22	42,64	1,43	1,44	36	
07:00:58	42,64	1,44	1,44	0	
07:59:24	42,64	1,44	1,44	9	
08:01:39	42,64	1,43	1,44	0	
08:24:00	42,66	1,43	1,44	13	Erhöhte Nachfrage aus Indien
08:59:10	42,66	1,43	1,44	36	
09:01:49	42,66	1,44	1,44	0	
09:59:10	42,66	1,44	1,44	12	
10:01:33	42,66	1,43	1,44	1	
10:12:29	42,72	1,43	1,44	12	Falkland-Krise
10:52:07	42,73	1,43	1,44	49	Ekuador-Krise
10:59:55	42,73	1,43	1,44	53	
11:01:20	42,73	1,44	1,44	0	
11:59:23	42,73	1,44	1,44	9	
12:01:24	42,73	1,43	1,44	0	
17:59:40	42,37	1,44	1,44	65	
18:01:12	42,37	1,45	1,44	0	
00:02:10	42,02	1,42	1,43	3	

Dieses Spielchen könnte man jetzt beliebig weit fortsetzen.

Ich habe die Simulation dann einige Zeit laufen lassen. Ein Screenshot auf der nächsten Seite zeigt mein Ergbnis.

5 Weiterführende Betrachtungen

Wie Sie mithilfe der Protokolle und des Quellcodes sehen funktioniert das Programm. Nun kann man sich also der Frage stellen welche der beiden Tankstellen auf Dauer mehr Gewinn einstreicht. Asso und Scholl nehmen sich beim Gewinn nicht viel. Mal fährt Asso den höheren Gewinn ein, mal Scholl. Allerdings ist Asso Scholl immer einen ein bisschen vorraus. Dies führt zur Vermutung das Asso auf Dauer mehr Gewinn macht als Scholl. Viele durchgeführte Simulationen bestätigen dies.
Asso könnte seinen Gewinn weiter optimieren indem die Tankstelle die Grenzen für die Preissenkung auf weniger als 25 stellt. Dadurch tanken im Verlauf der Simulation mehr Kunden bei Asso was folglich seinen Gewinn erhöht.

6 Quellcode

```
#include <stdlib.h>
#include <string.h>
#include <time.h>              // Für den Zufallsgenerator
#include <gtk/gtk.h>           // Den Gtk-Header implementieren. Ohne ihn läuft nichts ;-)

#define GUI "gui/tkm.xml"      // Das GUI wird mittels "gtk-builder-convert" von einer glade-Datei in einen XML-Text konvertiert

typedef struct{                // Die Hauptstruktur für das Programm --> besteht aus...

        GtkWidget       *win_main;              // ... Hauptfenster
        GtkWidget       *cmd_start;             // ... Button um die Simulation zu starten
        GtkWidget       *cmd_pause;             // ... Button um die Simulation zu pausieren
        GtkWidget       *cmd_reset;             // ... Button um die Simulation zurückzusetzen
        GtkLabel        *lbl_time;              // ... Label für die Uhrzeit in der Simulation
        GtkRange        *scl_speed;             // ... "Speedometer" ;-)
        GtkWidget       *txt_ausgabe;           // ... Element für Textausgabe
        GtkLabel        *lbl_oelpreis;          // ... Label für den akt. Ölpreis
        GtkLabel        *lbl_einkaufspreis;     // ... Label für den akt. Einkaufspreis (Ölpreis / 158,98)
        GtkLabel        *lbl_asso_preis;        // ... Label für den Preis bei Asso
        GtkLabel        *lbl_asso_kunden;       // ... Label für die Anzahl der Kunden bei Asso
        GtkLabel        *lbl_asso_kunden_std;   // ... Label für die Anzahl der Kunden bei Asso in der letzten Stunde
        GtkLabel        *lbl_scholl_preis;      // ... Label für den Preis bei Scholl
        GtkLabel        *lbl_scholl_kunden;     // ... Label für die Anzahl der Kunden bei Asso
        GtkLabel        *lbl_gewinn_asso;       // ... Label für den Gewinn von Asso
        GtkLabel        *lbl_gewinn_scholl;     // ... Label für den Gewinn von Scholl
        GtkLabel        *lbl_speed;             // ... Label für die Geschwindkeitsanzeige

} Tankomat;                    // Name für die Struktur

// ---------------------------- Alle globalen Variablen ----------------------

gint interval = 1000;                   // Timerinterval wird auf 1000 ms gesetzt
gint random_spotmarkt = 0;              // Eine Zufallszahl für den Spotmarkt
gint random_tanken = 0;                 // Eine Zufallszahl für die Tankstellen
gint timer = 0;                         // Der Timer für den Tankomat
gint tm_sek = 0;                        // Sekunden
gint tm_min = 0;                        // Minuten
gint tm_std = 0;                        // Stunden
gint tm_tag = 1;                        // Tag
gint rand_event = 0;                    // Zufallszahl für Ereignisse
gint rand_tanken_wo = 0;                // Zufalls für die Tankstelle wo die Kunden tanken
gint kunden_asso = 0;                   // Kundenzahl Asso
gint kunden_asso_std = 0;               // Alle Kunden von Asso der letzten Stunde
gint kunden_scholl = 0;                 // Kunden Scholl
```

```
gfloat akt_oelpreis = 67.85;              // Aktueller Ölpreis
gfloat event_preis_diff = 0;              // Geldbetrag der nach einem Ereignis auf den Ölpreis aufgeschlagen/abgezogen wird
gfloat akt_einkaufspreis = 0;             // Aktueller Einkaufspreis
gfloat einkaufspreis_asso = 0;
gfloat einkaufspreis_asso_puffer = 0;
gfloat einkaufspreis_scholl = 0;
const gfloat min_oel_steuer = 65;
const gfloat mwst = 19;
const gfloat oel_ebv = 0.46;
const gfloat marge_asso = 10;
const gfloat marge_scholl = 10;
gfloat benzin_preis_asso = 0;
gfloat benzin_preis_scholl = 0;
gfloat gewinn_asso = 0;
gfloat gewinn_scholl = 0;

gchar akt_zeit[100];                      // Die aktuelle Uhrzeit der Simulation
gchar text_ausgabe[100];                  // Ausgabe für das Textfeld
gchar akt_oelpreis_ausgabe[100];          // Ausgabe des aktuellen Ölpreises
gchar akt_einkaufspreis_ausgabe[100]; // Ausgabe des aktuellen Einkaufspreises
gchar event_ausgabe[100];
gchar benzin_preis_asso_ausgabe[100];
gchar benzin_preis_scholl_ausgabe[100];
gchar kunden_asso_ausgabe[100];
gchar kunden_asso_std_ausgabe[100];
gchar kunden_scholl_ausgabe[100];
gchar gewinn_asso_ausgabe[100];
gchar gewinn_scholl_ausgabe[100];

gboolean run_timer = FALSE;               // Läuft der Timer?
gboolean sim_pause = FALSE;               // Anwendung pausiert?
gboolean krieg_irak = FALSE;              // ist Krieg im Irak?
gboolean krieg_iran = FALSE;
gboolean krise_irak = FALSE;
gboolean krise_iran = FALSE;
gboolean krise_ekuador = FALSE;
gboolean krise_falkland = FALSE;

GdkColor color = {0, 0xEEFF, 0xEBFF, 0xE6FF};  // Hintergrundfarbe für win_main

GtkTextBuffer *buffer;                    // Für Ausgabe der Ereignisse

// --------------------------- Funktionsprototypen ---------------------------

int main (int argc, char *argv[]);        // Hauptfunktion

void on_win_main_destroy (GtkObject *object, Tankomat *program);   // Funktion für das Beenden des Programmes
void on_cmd_start_clicked (GtkObject *object, Tankomat *program);   // Starten der Simulation
```

```
void on_Timer_Func(Tankomat *program);                          // Alles was innerhalb eines Intervals passieren soll...
void on_cmd_pause_clicked (GtkObject *object, Tankomat *program); // Simulation pausieren
void on_cmd_reset_clicked(GtkObject *object, Tankomat *program);   // Simulation zurücksetzen
void on_scl_speed_change_value (GtkObject *object, Tankomat *program);       // Funktion zum ändern des Intervals
void on_event_show_text (gchar *text);
void ov_event_show_oelpreis (Tankomat *program);
void show_time(Tankomat *program);
void on_event_transportkosten (gboolean steigen);
void on_event_krisen (gchar *gebiet, gboolean krise_aktiv, gboolean steigen);
void on_event_kriege (gchar *gebiet, gboolean krieg_aktiv, gboolean steigen);
void on_event_nachfrage (gchar *gebiet, gboolean steigen);
void on_event_opec(gboolean steigen);
void on_event_wechselkurs(gboolean steigen);
void calc_preis_asso(Tankomat *program);
void akt_kunden_asso(Tankomat *program);
void akt_kunden_scholl(Tankomat *program);
void calc_preis_scholl(Tankomat *program);
void calc_gewinn_asso(Tankomat *program);
void calc_gewinn_scholl(Tankomat *program);

gboolean build_app (Tankomat *program);                         // Funktion zum Erstellen der Applikation

// -------------------------- Funktionen ----------------------------------

gboolean build_app (Tankomat *program){                         // Funktion zum Erstellen der Applikation

    GtkBuilder *builder;
        builder = gtk_builder_new ();
    gtk_builder_add_from_file (builder, GUI, NULL);

    program->win_main = GTK_WIDGET (gtk_builder_get_object (builder, "win_main"));
    program->cmd_start = GTK_WIDGET (gtk_builder_get_object (builder, "cmd_start"));
    program->cmd_pause = GTK_WIDGET (gtk_builder_get_object (builder, "cmd_pause"));
    program->cmd_reset = GTK_WIDGET (gtk_builder_get_object (builder, "cmd_reset"));
    program->lbl_time = GTK_LABEL (gtk_builder_get_object (builder, "lbl_time"));
    program->scl_speed = GTK_RANGE (gtk_builder_get_object (builder, "scl_speed"));
    program->txt_ausgabe = GTK_WIDGET (gtk_builder_get_object (builder, "txt_ausgabe"));
    program->lbl_oelpreis = GTK_LABEL (gtk_builder_get_object (builder, "lbl_oelpreis"));
    program->lbl_einkaufspreis = GTK_LABEL (gtk_builder_get_object (builder, "lbl_einkaufspreis"));
    program->lbl_asso_preis = GTK_LABEL (gtk_builder_get_object (builder, "lbl_asso_preis"));
    program->lbl_asso_kunden = GTK_LABEL (gtk_builder_get_object (builder, "lbl_asso_kunden"));
    program->lbl_asso_kunden_std = GTK_LABEL (gtk_builder_get_object (builder, "lbl_asso_kunden_std"));
    program->lbl_scholl_preis = GTK_LABEL (gtk_builder_get_object (builder, "lbl_scholl_preis"));
    program->lbl_scholl_kunden = GTK_LABEL (gtk_builder_get_object (builder, "lbl_scholl_kunden"));
    program->lbl_gewinn_asso = GTK_LABEL (gtk_builder_get_object (builder, "lbl_gewinn_asso"));
    program->lbl_gewinn_scholl = GTK_LABEL (gtk_builder_get_object (builder, "lbl_gewinn_scholl"));
    program->lbl_speed = GTK_LABEL (gtk_builder_get_object (builder, "lbl_speed"));
```

```
        gtk_widget_modify_bg(program->win_main, GTK_STATE_NORMAL, &color);
        buffer = gtk_text_view_get_buffer (GTK_TEXT_VIEW (program->txt_ausgabe));

    gtk_builder_connect_signals (builder, program);
        g_object_unref (G_OBJECT (builder));

    return TRUE;
}

int main (int argc, char *argv[]) {                    // *** Hauptfunktion

    Tankomat *program;                                 // Ableitung der Struktur erstellen
    program = g_slice_new (Tankomat);                  // Speicher für die Struktur anfordern

        srand( time(NULL) );                           // Zufallsgenerator initialisieren

    gtk_init (&argc, &argv);                           // GTK+ initialisieren

    if (build_app (program) == FALSE) return 1;        // War das Erstellen der Applikation erfolgreich?

    gtk_widget_show (program->win_main);               // Das Hauptfenster (win_main) anzeigen

    gtk_main ();                                       // Den Gtk-Event-Loop ausführen

        if(run_timer == TRUE) gtk_timeout_remove(timer);// Den Timer entfernen

    g_slice_free (Tankomat, program);                  // Die Speicherbereiche die belegt waren wieder freigeben

    return 0;
}

void on_win_main_destroy (GtkObject *object, Tankomat *program){  // *** Funktion für das Beenden des Programmes

        gtk_main_quit();                               // Hier wird die Funktion zum Beenden der Applikation aufgerufen

        return;
}

void on_cmd_start_clicked (GtkObject *object, Tankomat *program){  // *** Hier wird die Simulation gestartet

        timer = gtk_timeout_add (interval, (GtkFunction) on_Timer_Func, program);      //Starten des Timers
        gtk_widget_set_sensitive(program->cmd_pause, TRUE);
        gtk_widget_set_sensitive(program->cmd_start, FALSE);

        show_time(program);

        if(sim_pause == FALSE)
                on_event_show_text("Simulation gestartet\t\t[OK]");
        else
```

```
		on_event_show_text("Simulation wird fortgesetzt\t[OK]");

	ov_event_show_oelpreis(program);

	if(sim_pause == FALSE){
		einkaufspreis_asso = akt_einkaufspreis;
		einkaufspreis_asso_puffer = akt_einkaufspreis;
		einkaufspreis_scholl = akt_einkaufspreis; }

	benzin_preis_asso = (einkaufspreis_asso + min_oel_steuer + ((akt_einkaufspreis + min_oel_steuer) / 100 * mwst) +
marge_asso + oel_ebv) / 100;
	benzin_preis_scholl = (einkaufspreis_scholl + 5 + min_oel_steuer + ((akt_einkaufspreis + min_oel_steuer) / 100 * mwst) +
marge_scholl + oel_ebv) / 100;

	sprintf(benzin_preis_asso_ausgabe, "Aktueller Preis: %1.2f Euro", benzin_preis_asso);
	sprintf(benzin_preis_scholl_ausgabe, "Aktueller Preis: %1.2f Euro", benzin_preis_scholl);

	gtk_label_set_markup(program->lbl_asso_preis, benzin_preis_asso_ausgabe);
	gtk_label_set_markup(program->lbl_scholl_preis, benzin_preis_scholl_ausgabe);

	sim_pause = FALSE;
	run_timer = TRUE;

	return;
}

void on_Timer_Func(Tankomat *program){			// *** Pro Timerinterval wird diese Funktion ausgeführt

	tm_sek++;					// Uhrzeit vorantreiben

	if(tm_sek == 60){
		tm_sek = 0;
		tm_min++; }
	if(tm_min == 60){				// Nach 60 Minuten...
		if(tm_std % 6 == 0){			// Ölpreis beim Spotmarkt abfragen und dementsprechend reagieren
			einkaufspreis_asso_puffer = akt_einkaufspreis;
			einkaufspreis_scholl = akt_einkaufspreis; }
		tm_min = 0;
		tm_std++;
		calc_preis_asso(program);		// Preis für Asso berechnen
		calc_preis_scholl(program); }		// Preis für Scholl berechnen
	if(tm_std == 24){
		tm_std = 0;
		tm_tag++; }

	show_time(program);

	rand_event = (rand () % 100) + 1;
```

```
if(rand_event == 36){                          // Mit 1 prozentiger Wahrscheinlichkeit wird ein Ereignis ausgelöst
        random_spotmarkt = (rand () % 1000) + 1;
        if(random_spotmarkt == 10) on_event_transportkosten(TRUE);
        else if(random_spotmarkt == 20) on_event_transportkosten(FALSE);
        else if(random_spotmarkt == 30) on_event_krisen("Irak", krise_irak, TRUE);
        else if(random_spotmarkt == 40) on_event_krisen("Iran", krise_iran, TRUE);
        else if(random_spotmarkt == 50) on_event_krisen("Ekuador", krise_ekuador, TRUE);
        else if(random_spotmarkt == 60) on_event_krisen("Falkland", krise_falkland, TRUE);
        else if(random_spotmarkt == 70) on_event_krisen("Irak", krise_irak, FALSE);
        else if(random_spotmarkt == 80) on_event_krisen("Iran", krise_iran, FALSE);
        else if(random_spotmarkt == 90) on_event_krisen("Ekuador", krise_ekuador, FALSE);
        else if(random_spotmarkt == 100) on_event_krisen("Falkland", krise_falkland, FALSE);
        else if(random_spotmarkt == 110) on_event_kriege("Irak", krieg_irak, TRUE);
        else if(random_spotmarkt == 120) on_event_kriege("Iran", krieg_iran, TRUE);
        else if(random_spotmarkt == 130) on_event_kriege("Irak", krieg_irak, FALSE);
        else if(random_spotmarkt == 140) on_event_kriege("Iran", krieg_iran, FALSE);
        else if(random_spotmarkt == 150) on_event_nachfrage("China", TRUE);
        else if(random_spotmarkt == 160) on_event_nachfrage("Indien", TRUE);
        else if(random_spotmarkt == 170) on_event_nachfrage("Brasilien", TRUE);
        else if(random_spotmarkt == 180) on_event_nachfrage("China", FALSE);
        else if(random_spotmarkt == 190) on_event_nachfrage("Indien", FALSE);
        else if(random_spotmarkt == 200) on_event_nachfrage("Brasilien", FALSE);
        else if(random_spotmarkt == 210) on_event_opec(TRUE);
        else if(random_spotmarkt == 220) on_event_opec(FALSE);
        else if(random_spotmarkt == 230) on_event_wechselkurs(TRUE);
        else if(random_spotmarkt == 240) on_event_wechselkurs(FALSE);

        ov_event_show_oelpreis(program);       // Zeige den aktuellen Ölpreis an
}

random_tanken = (rand () % 60) + 1;

if(random_tanken == 30){                       // Mit einer Wahrscheinlichkeit von 1 / 60 tankt ein vorbeifahrendes Auto

        rand_tanken_wo = (rand () % 10) + 1;

        if((rand_tanken_wo >= 1) && (rand_tanken_wo <= 6)){        // 60% der tanken günstig

                if(benzin_preis_asso < benzin_preis_scholl){
                        kunden_asso++;
                        kunden_asso_std++;
                        akt_kunden_asso(program);
                        calc_gewinn_asso(program); }
                else if(benzin_preis_asso > benzin_preis_scholl){
                        kunden_scholl++;
                        akt_kunden_scholl(program);
                        calc_gewinn_scholl(program); }
        }
```

```
                if((rand_tanken_wo == 7) || (rand_tanken_wo == 8)){        // 20% der Tankenden biegen zu Asso ein

                        kunden_asso++;
                        kunden_asso_std++;
                        akt_kunden_asso(program);
                        calc_gewinn_asso(program);
                }

                if((rand_tanken_wo == 9) || (rand_tanken_wo == 10)){       // 20% der Tankenden biegen zu Scholl ein

                        kunden_scholl++;
                        akt_kunden_scholl(program);
                        calc_gewinn_scholl(program);
                }
        }

        return;
}

void on_cmd_pause_clicked (GtkObject *object, Tankomat *program){          // *** Funktion für das Pausieren des Timers

        sim_pause = TRUE;
        gtk_timeout_remove(timer);
        gtk_widget_set_sensitive(program->cmd_pause, FALSE);
        gtk_widget_set_sensitive(program->cmd_start, TRUE);
        on_event_show_text("Simulation unterbrochen\t[OK]");
        run_timer = FALSE;

        return;
}

void on_cmd_reset_clicked(GtkObject *object, Tankomat *program){

        if(run_timer == TRUE) gtk_timeout_remove(timer);
        sim_pause = FALSE;
        run_timer = FALSE;
        gtk_widget_set_sensitive(program->cmd_pause, FALSE);
        gtk_widget_set_sensitive(program->cmd_start, TRUE);
        tm_sek = 0;
        tm_min = 0;
        tm_std = 0;
        tm_tag = 1;
        kunden_asso = 0;
        kunden_asso_std = 0;
        kunden_scholl = 0;
        akt_oelpreis = 67.85;
        krieg_irak = FALSE;
        krieg_iran = FALSE;
        krise_irak = FALSE;
```

```
	krise_iran = FALSE;
	krise_ekuador = FALSE;
	krise_falkland = FALSE;
	benzin_preis_asso = 0;
	benzin_preis_scholl = 0;
	gewinn_asso = 0;
	gewinn_scholl = 0;
	gtk_text_buffer_set_text (buffer, "", -1);
	akt_kunden_asso(program);
	akt_kunden_scholl(program);
	calc_gewinn_asso(program);
	calc_gewinn_scholl(program);
	ov_event_show_oelpreis(program);
	sprintf(benzin_preis_asso_ausgabe, "Aktueller Preis: %1.2f Euro", benzin_preis_asso);
	sprintf(benzin_preis_scholl_ausgabe, "Aktueller Preis: %1.2f Euro", benzin_preis_scholl);
	gtk_label_set_markup(program->lbl_asso_preis, benzin_preis_asso_ausgabe);
	gtk_label_set_markup(program->lbl_scholl_preis, benzin_preis_scholl_ausgabe);
	show_time(program);
}

void on_scl_speed_change_value (GtkObject *object, Tankomat *program){	// *** Funktion zum ändern des Intervals

	gdouble akt_value;

	akt_value = gtk_range_get_value (program->scl_speed);		// aktuellen Wert des Schiebereglers auslesen

	if(akt_value == 1){
		gtk_label_set_markup(program->lbl_speed, "Echtzeit"); interval = 1000; }
	else if(akt_value == 2){
		gtk_label_set_markup(program->lbl_speed, "Schnell"); interval = 100; }
	else if(akt_value == 3){
		gtk_label_set_markup(program->lbl_speed, "Sehr schnell"); interval = 5; }
	else if(akt_value == 4){
		gtk_label_set_markup(program->lbl_speed, "Ultraschnell"); interval = 1; }

	if(run_timer == TRUE){			// alten Timer entfernen und neuen setzen aber nur wenn der Timer bereits läuft
		gtk_timeout_remove(timer);
		timer = gtk_timeout_add (interval, (GtkFunction) on_Timer_Func, program);
	}

	return;
}

void on_event_show_text (gchar *text){		// *** Funktion zum Ausgeben von Text

	sprintf (text_ausgabe, "[%.4d - %.2d:%.2d:%.2d] - %s\n",tm_tag, tm_std, tm_min, tm_sek, text);
	gtk_text_buffer_insert_at_cursor(buffer, text_ausgabe, -1);

	return; }
```

```
void ov_event_show_oelpreis (Tankomat *program){

	sprintf (akt_oelpreis_ausgabe, "<big><big><span foreground=\"#ff0000\">Ölpreis: %3.2f Euro je Barrel</span></big></big>",
akt_oelpreis);
	gtk_label_set_markup(program->lbl_oelpreis, akt_oelpreis_ausgabe);

	akt_einkaufspreis = (akt_oelpreis / 158.98) * 100;

	sprintf (akt_einkaufspreis_ausgabe, "<big><big><span foreground=\"#008312\">Einkaufspreis: %3.2f Cent je
Liter</span></big></big>", akt_einkaufspreis);
	gtk_label_set_markup(program->lbl_einkaufspreis, akt_einkaufspreis_ausgabe);

	return;
}

void show_time(Tankomat *program){

	sprintf (akt_zeit, "<big><big><big><span foreground=\"#000000\">Uhrzeit: %.2d:%.2d:
%.2d</span></big></big></big>",tm_std, tm_min, tm_sek);
	gtk_label_set_markup(program->lbl_time, akt_zeit);

	return;
}

void on_event_transportkosten (gboolean steigen){		// Event Transportkosten

	event_preis_diff = (rand () % 10) + 1;
	event_preis_diff = event_preis_diff / 10;

	if(steigen)sprintf(event_ausgabe,"Transportkosten steigen um %1.1f Cent", event_preis_diff);
	else		sprintf(event_ausgabe,"Transportkosten sinken um %1.1f Cent", event_preis_diff);

	on_event_show_text(event_ausgabe);
	event_preis_diff = event_preis_diff / 100;

	if(steigen)		akt_oelpreis = akt_oelpreis + event_preis_diff;
	else		akt_oelpreis = akt_oelpreis - event_preis_diff;

	return;
}

void on_event_krisen (gchar *gebiet, gboolean krise_aktiv, gboolean steigen){	// Event Krisen

	if(steigen){

		event_preis_diff = (rand () % 100) + 1;
		event_preis_diff = event_preis_diff / 10;
```

```
		if(krise_aktiv == FALSE){
			sprintf(event_ausgabe,"%s-Krise: Ölpreis steigt um %1.1f Cent",gebiet, event_preis_diff);
			if(strcmp(gebiet,"Irak")== 0)		krise_irak = TRUE;
			else if(strcmp(gebiet,"Iran") == 0)	krise_iran = TRUE;
			else if(strcmp(gebiet,"Ekuador") == 0)	krise_ekuador = TRUE;
			else if(strcmp(gebiet,"Falkland")== 0)	krise_falkland = TRUE;
		}
		else	sprintf(event_ausgabe,"%s-Krise verschärft sich: Ölpreis steigt um %1.1f Cent",gebiet, event_preis_diff);

		on_event_show_text(event_ausgabe);
		event_preis_diff = event_preis_diff / 100;
		akt_oelpreis = akt_oelpreis + event_preis_diff;
	}

	else{

		event_preis_diff = (rand () % 700) + 1;
		event_preis_diff = event_preis_diff / 10;

		if(krise_aktiv == TRUE){
			sprintf(event_ausgabe,"%s-Krise entspannt sich: Ölpreis fällt um %1.1f Cent",gebiet, event_preis_diff);
			if(strcmp(gebiet,"Irak")== 0)		krise_irak = FALSE;
			else if(strcmp(gebiet,"Iran") == 0)	krise_iran = FALSE;
			else if(strcmp(gebiet,"Ekuador") == 0)	krise_ekuador = FALSE;
			else if(strcmp(gebiet,"Falkland")== 0)	krise_falkland = FALSE;

			on_event_show_text(event_ausgabe);
			event_preis_diff = event_preis_diff / 100;
			akt_oelpreis = akt_oelpreis - event_preis_diff;
		}
	}

	return;
}

void on_event_kriege (gchar *gebiet, gboolean krieg_aktiv, gboolean steigen){   // Event Kriege

	if(steigen){

		event_preis_diff = (rand () % 1000) + 1;
		event_preis_diff = event_preis_diff / 10;

		if(krieg_aktiv == FALSE){

			sprintf(event_ausgabe, "Krieg im %s. Ölpreis steigt um %1.1f Cent", gebiet, event_preis_diff);
			if(strcmp(gebiet,"Irak")== 0)		krieg_irak = TRUE;
			else if(strcmp(gebiet,"Iran") == 0)	krieg_iran = TRUE;
		}
		else	sprintf(event_ausgabe, "Ölplattform explodiert im %s-Krieg. Ölpreis steigt um %1.1f Cent", gebiet,
```

```
event_preis_diff);

                on_event_show_text(event_ausgabe);
                event_preis_diff = event_preis_diff / 100;
                akt_oelpreis = akt_oelpreis + event_preis_diff;
        }

        else{

                event_preis_diff = (rand () % 1000) + 1;
                event_preis_diff = event_preis_diff / 10;

                if(krieg_aktiv){
                        sprintf(event_ausgabe,"%s-Krieg beendet. Ölpreis fällt um %1.1f Cent",gebiet, event_preis_diff);
                        if(strcmp(gebiet,"Irak")== 0)           krieg_irak = FALSE;
                        else if(strcmp(gebiet,"Iran") == 0)     krieg_iran = FALSE;

                        on_event_show_text(event_ausgabe);
                        event_preis_diff = event_preis_diff / 100;
                        akt_oelpreis = akt_oelpreis - event_preis_diff;
                }
        }

        return;
}

void on_event_nachfrage (gchar *gebiet, gboolean steigen){          // Event Nachfrage

        event_preis_diff = (rand () % 100) + 1;
        event_preis_diff = event_preis_diff / 10;

        if(steigen){

                sprintf(event_ausgabe,"Erhöhte Nachfrage aus %s: Ölpreis steigt um %1.1f Cent",gebiet, event_preis_diff);

                on_event_show_text(event_ausgabe);
                event_preis_diff = event_preis_diff / 100;
                akt_oelpreis = akt_oelpreis + event_preis_diff;
        }

        else{
                sprintf(event_ausgabe,"Geringere Nachfrage aus %s: Ölpreis fällt um %1.1f Cent",gebiet, event_preis_diff);

                on_event_show_text(event_ausgabe);
                event_preis_diff = event_preis_diff / 100;
                akt_oelpreis = akt_oelpreis - event_preis_diff;
        }

        return;
```

```
}

void on_event_opec(gboolean steigen){              // Event OPEC

        event_preis_diff = (rand () % 100) + 1;
        event_preis_diff = event_preis_diff / 10;

        if(steigen){

                sprintf(event_ausgabe,"OPEC prognostiziert steigende Preise. Ölpreis steigt um %1.1f Cent", event_preis_diff);

                on_event_show_text(event_ausgabe);
                event_preis_diff = event_preis_diff / 100;
                akt_oelpreis = akt_oelpreis + event_preis_diff;
        }

        else{
                sprintf(event_ausgabe,"OPEC prognostiziert sinkende Preise. Ölpreis fällt um %1.1f Cent", event_preis_diff);

                on_event_show_text(event_ausgabe);
                event_preis_diff = event_preis_diff / 100;
                akt_oelpreis = akt_oelpreis - event_preis_diff;
        }

        return;
}

void on_event_wechselkurs(gboolean steigen){    // Event Wechselkurs

        event_preis_diff = (rand () % 10) + 1;
        event_preis_diff = event_preis_diff / 10;

        if(steigen){

                sprintf(event_ausgabe,"Dollarkurs schwächelt. Ölpreis fällt um %1.1f Cent", event_preis_diff);

                on_event_show_text(event_ausgabe);
                event_preis_diff = event_preis_diff / 100;
                akt_oelpreis = akt_oelpreis - event_preis_diff;
        }

        else{
                sprintf(event_ausgabe,"Dollarkurs wird stärker. Ölpreis steigt um %1.1f Cent", event_preis_diff);

                on_event_show_text(event_ausgabe);
                event_preis_diff = event_preis_diff / 100;
                akt_oelpreis = akt_oelpreis + event_preis_diff;
        }
```

```
	return;
}

void calc_preis_asso(Tankomat *program){	// Neuen Preis für Asso berechnen

	if(kunden_asso_std > 35)				// Bei mehr als 35 Kunden...
		einkaufspreis_asso = einkaufspreis_asso + 1;
	else if(kunden_asso_std < 25)
		einkaufspreis_asso = einkaufspreis_asso - 1;	// Bei weniger als 25 Kunden...

	if(einkaufspreis_asso < einkaufspreis_asso_puffer)
		einkaufspreis_asso = einkaufspreis_asso_puffer;

	benzin_preis_asso = (einkaufspreis_asso + min_oel_steuer + ((akt_einkaufspreis + min_oel_steuer) / 100 * mwst) +
marge_asso + oel_ebv) / 100;

	sprintf(benzin_preis_asso_ausgabe, "Aktueller Preis: %1.2f Euro", benzin_preis_asso);

	gtk_label_set_markup(program->lbl_asso_preis, benzin_preis_asso_ausgabe);

	kunden_asso_std = 0;

	return;
}

void akt_kunden_asso(Tankomat *program){	// Kundenzahl von Asso anzeigen

	sprintf(kunden_asso_ausgabe, "Anzahl Kunden: %d", kunden_asso);
	sprintf(kunden_asso_std_ausgabe, "Anzahl Kunden letzte Stunde: %d", kunden_asso_std);

	gtk_label_set_markup(program->lbl_asso_kunden, kunden_asso_ausgabe);
	gtk_label_set_markup(program->lbl_asso_kunden_std, kunden_asso_std_ausgabe);

	return;
}

void akt_kunden_scholl(Tankomat *program){	// Kundenzahl von Scholl anzeigen

	sprintf(kunden_scholl_ausgabe, "Anzahl Kunden: %d", kunden_scholl);
	gtk_label_set_markup(program->lbl_scholl_kunden, kunden_scholl_ausgabe);

	return;
}

void calc_preis_scholl(Tankomat *program){	// Berechne Preis für Scholl

	benzin_preis_scholl = (einkaufspreis_scholl + 5 + min_oel_steuer + ((akt_einkaufspreis + min_oel_steuer) / 100 * mwst) +
marge_scholl + oel_ebv) / 100;
	sprintf(benzin_preis_scholl_ausgabe, "Aktueller Preis: %1.2f Euro", benzin_preis_scholl);
```

```
	gtk_label_set_markup(program->lbl_scholl_preis, benzin_preis_scholl_ausgabe);

	return;
}

void calc_gewinn_asso(Tankomat *program){	// Gewinn von Asso kalkulieren

	gewinn_asso = gewinn_asso + benzin_preis_asso;
	sprintf(gewinn_asso_ausgabe, "Gewinn: %.2f Euro", gewinn_asso);
	gtk_label_set_markup(program->lbl_gewinn_asso, gewinn_asso_ausgabe);

	return;
}

void calc_gewinn_scholl(Tankomat *program){	// Gewinn von Scholl kalkulieren

	gewinn_scholl = gewinn_scholl + benzin_preis_scholl;
	sprintf(gewinn_scholl_ausgabe, "Gewinn: %.2f Euro", gewinn_scholl);
	gtk_label_set_markup(program->lbl_gewinn_scholl, gewinn_scholl_ausgabe);

	return;
}
```

Anlage A – Teilprobleme und deren Lösungsidee

Teilproblem	Mögliche Lösung
Simulation ablaufen lassen	Implementierung eines Timers
Startwert des Ölpreises	Recherche im Internet und entsprechende Implementierung
Simulation des Ölpreises	Auslösen von zufallsbasierenden Ereignissen
Startwert des Preises an den Tankstellen	Ölpreis + Aufschläge (MwSt, MinöSTV, EBV, Marge)
Entscheidung ob Auto tankt	Bestimmung eines Zufallswertes
Entscheidung wo Auto tankt (Preisgünstig)	Bestimmung eines Zufallswertes
Entscheidung wo Auto tankt (Stammkunden)	Bestimmung eines Zufallswertes
Anzahl der Kunden bestimmen	Tankstellenspezifischen Wert inkrementieren sobald Auto tankt
Anzahl der Kunden letzte Stunde (Asso)	Anzahl Kunden bestimmen → Wert nach einer Stunde Null setzen
Preisfestlegung Asso	Jede Stunde Wert „Anzahl Kunden letzte Stunde“ prüfen. Bei > 35 um einen Cent erhöhen. Bei < 25 um einen Cent verringern. Sonst nichts machen. Puffer benutzen um diesen Einkaufspreis mit dem aktuellen Ölpreis zu vergleichen. Puffer alle 6 Stunden mit dem aktuellen Preis vom Spotmarkt überschreiben.
Preisfestlegung Scholl	Alle 6 Stunden (und am Anfang) den Ölpreis vom Spotmarkt abfragen und 5 Cent addieren.

Anlage B –Ereignisse auf dem Spotmarkt Öl

Zahl	Ereignis	Veränderungen des Ölpreises
10	Transportkosten steigen	+ 0,1 bis 1,0 Cent
20	Transportkosten sinken	- 0,1 bis 1,0 Cent
30	Krise im Irak	+ 0,1 bis 10,0 Cent
40	Krise im Iran	+ 0,1 bis 10,0 Cent
50	Krise in Ekuador	+ 0,1 bis 10,0 Cent
60	Falklandkrise	+ 0,1 bis 10.0 Cent
30 (falls schon aktiv)	Irak-Krise verschärft sich	+ 0,1 bis 10,0 Cent
40 (falls schon aktiv)	Iran-Krise verschärft sich	+ 0,1 bis 10,0 Cent
50 (falls schon aktiv)	Ekuador-Krise verschärft sich	+ 0,1 bis 10,0 Cent
60 (falls schon aktiv)	Falkland-Krise verschärft sich	+ 0,1 bis 10,0 Cent
70	Irak-Krise entspannt sich	- 0,1 bis 70,0 Cent
80	Iran-Krise entspannt sich	- 0,1 bis 70,0 Cent
90	Ekuador-Krise entspannt sich	- 0,1 bis 70,0 Cent
100	Falkland-Krise entspannt sich	- 0,1 bis 70,0 Cent
110	Krieg im Irak	+ 0,1 bis 100,0 Cent
120	Krieg im Iran	+ 0,1 bis 100,0 Cent
110 (falls schon aktiv)	Ölplattform explodiert im Irak-Krieg	+ 0,1 bis 100,0 Cent
120 (falls schon aktiv)	Ölplattform explodiert im Iran-Krieg	+ 0,1 bis 100,0 Cent
130	Irak-Krieg beendet	- 0,1 bis 100,0 Cent
140	Iran-Krieg beendet	- 0,1 bis 100,0 Cent
150	Erhöhte Nachfrage aus China	+ 0,1 bis 10,0 Cent
160	Erhöhte Nachfrage aus Indien	+ 0,1 bis 10,0 Cent
170	Erhöhte Nachfrage aus Brasilien	+ 0,1 bis 10,0 Cent
180	Geringere Nachfrage aus China	- 0,1 bis 10,0 Cent
190	Geringere Nachfrage aus Indien	- 0,1 bis 10,0 Cent
200	Geringere Nachfrage aus Brasilien	- 0,1 bis 10,0 Cent
210	OPEC prognostiziert steigende Ölpreise	+ 0,1 bis 10,0 Cent
220	OPEC prognostiziert sinkende Ölpreise	- 0,1 bis 10,0 Cent
230	Dollarkurs schwächelt	- 0,1 bis 1,0 Cent
240	Dollarkurs steigt	+ 0,1 bis 1,0 Cent

Anlage C –Funktionen und deren Wirkungsweise

int main (int argc, char *argv[])	Hauptfunktion für das Programm. Hier wird GTK initialisiert, der Zufallsgenerator gestartet, die Oberfläche der Anwendung erstellt und die Kontrolle der Gtk_Main_Loop übergeben.
void on_win_main_destroy (GtkObject *object, Tankomat *program)	Funktion zum Beenden der Anwendung
void on_cmd_start_clicked (GtkObject *object, Tankomat *program)	Funktion zum Starten der Simulation
void on_Timer_Func(Tankomat *program)	Der Inhalt dieser Funktion wird pro Timerinterval einmal aufgerufen
void on_cmd_pause_clicked (GtkObject *object, Tankomat *program)	Funktion zum Pausieren der Simulation
void on_cmd_reset_clicked(GtkObject *object, Tankomat *program)	Funktion zum Zurücksetzen der Simulation
void on_scl_speed_change_value (GtkObject *object, Tankomat *program)	Funktion zum Ändern des Timerintervals (Echtzeit, Schnell, Sehr schnell, Ultraschnell)
void on_event_show_text (gchar *text)	Funktion zum Anzeigen von Eventtext in dem Textfeld
void ov_event_show_oelpreis (Tankomat *program)	Funktion zum Anzeigen des Ölpreises. Einmal in Euro pro Barrel und Cent pro Liter
void show_time(Tankomat *program)	Funktion zum Anzeigen der Uhrzeit im Programm
void on_event_transportkosten (gboolean steigen)	Funktion zum Auswerten ob Transportkosten steigen oder fallen
void on_event_krisen (gchar *gebiet, gboolean krise_aktiv, gboolean steigen)	Funktion zum Auswerten der Krisen-Ereignisse
void on_event_kriege (gchar *gebiet, gboolean krieg_aktiv, gboolean steigen)	Funktion zum Auswerten der Kriegs-Ereignisse
void on_event_nachfrage (gchar *gebiet, gboolean steigen)	Funktion zum Auswerten der Nachfrage-Ereignisse
void on_event_opec(gboolean steigen)	Funktion zum Auswerten der Preisprognostizierungen
void on_event_wechselkurs(gboolean steigen)	Funktion zum Auswerten von Wechselkursereignissen
void calc_preis_asso(Tankomat *program)	Berechnet den neuen Preis von Asso

void akt_kunden_asso(Tankomat *program)	Zeigt alle Kunden von Asso an
void akt_kunden_scholl(Tankomat *program)	Zeigt alle Kunden von Scholl an
void calc_preis_scholl(Tankomat *program)	Berechnet den neuen Preis von Scholl
void calc_gewinn_asso(Tankomat *program)	Berechnet den Gewinn von Asso
void calc_gewinn_scholl(Tankomat *program)	Berechnet den Gewinn von Scholl

Anlage D –Wichtige Variablen und deren Funktion

Datentyp	Variable	Funktion
gint	interval	Wird benutzt um das Timerinterval festzulegen
gint	spotmarkt	Wird benutzt um später zwischen Ereignissen zu differenzieren
gint	random_tanken	Wird benutzt um zu entscheiden ob ein Auto tankt
gint	tm_sek, tm_min, tm_std, tm_tag	Variablen für die aktuelle Uhrzeit und den Tag
gint	rand_event	Durch diese Variable wird entschieden ob eine weitere Zufallszahl bestimmt werden soll um ein Ereignis auszulösen
gint	rand_tanken_wo	Legt fest an welcher Tankstelle getankt wird
gint	kunden_asso	Speichert die Anzahl der Kunden von Asso
gint	kunden_scholl	Speichert die Anzahl der Kunden von Scholl
gint	kunden_asso_std	Anzahl der Kunden letzte Stunde (Asso)
gfloat	akt_oelpreis	Aktueller Ölpreis am Spotmarkt. Wird mit 67,85 Euro pro Barrel initialisiert
gfloat	event_preis_diff	Geldbetrag der innerhalb eines Events bestimmt wird. Wird dann auf den Ölpreis aufgeschlagen oder von selbigem abgezogen
gfloat	akt_einkaufspreis	Aktueller Ölpreis am Spotmarkt in Cent pro Liter. Dieser Preis wird von den Tankstellen benutzt
gfloat	benzin_preis_asso	Benzinpreis bei Asso
gfloat	benzin_preis_scholl	Benzinpreis bei Scholl
gboolean	run_timer	Verschiedene Aufgaben für den Timer
gboolean	sim_pause	Wird benutzt um die Simulation zu Pausieren
GdkColor	color	Hintergrundfarbe für die Anwendung
GtkTextBuffer	buffer	Wichtig für die Ausgabe von Ereignissen im Textfeld
(struct)	Tankomat	Struktur für alle GUI-Komponenten in der Anwendung

Anlage E - Nassi-Shneiderman-Diagramm