Aufgabe 2: Trau, schau, wem!

Lösungsidee:

Der Programmierauftrag bestand darin einen CLUBSTER Agenten und einen ULI Agenten zu programmieren, die über ein Kommunikationsprotokoll miteinander kommunizieren. Beim ersten lesen dachte ich an ein Client-Programm und ein Server Programm, wobei der Server ULIs Aufgaben übernimmt. Da dies laut FAQ nicht nötig ist, habe ich alles ein Programm gepackt. Der ULI Teil übernimmt dabei nicht nur die EVE und PEVE Nachrichten, sondern übermittelt die einzelnen SMS Nachrichten zwischen den Clubstern.

Mein Kommunikationsprotokoll soll dabei so funktionieren:

Mein Datenträger soll dabei eine Variable sein, welche die Nummer des Sender/Empfänger Clubsters enthält und zwei weiteren Zahlen, die für Ort und Zeit stehen. In einem vierten Parameter wird festgehalten, um was für einen Nachrichtentyp (WUW, EVE, PEVE) es sich überhaupt handelt. Um einen Tagesablauf zu simulieren, sollen die Clubsters ihre empfangenen ungelesenen Nachrichten und die zu sendenden Nachrichten in zwei Buffern speichern. Wenn ein neuer Tag beginnt und ein Clubster nichts vom Treffpunkt erfahren hat, dann denkt er sich nach seiner Strategie eine gewisse Anzahl von Fragen aus und schreibt sie in seinen Sendebuffer. Damit aber ein Clubster dadurch nicht alle Nachrichten verschicken kann, liest ULI immer nur eine Nachricht aus dem Buffer. Wenn ULI die Nachrichten von einem Clubster zum anderen übermittelt, landet sie im Buffer des Clubsters mit der in der Nachricht gespeicherten Nummer, wobei diese Nummer zuvor mit der des Absenders überschrieben wird, damit dieser auf weis wohin er Antworten soll.

Die Clubsters sollen dabei Unterobjekte von ULI sein, die sich nicht kennen und von ULI erschaffen werden. So können über den ULI Programmteil die kompletten Frage- und Antwortstrategien bestimmt, ihre Sympathiewerte zufällig vergeben und die Verlässigkeitswerte zurückgesetzt werden.

Intern sollen die Clubsters so arbeiten:

In gewissen Zeitintervallen sollen sie eine Nachricht aus dem Buffer nehmen und sie je nach Typ verarbeiten. WUWs werden wenn möglich beantwortet, wobei zuvor der Fragende nach Verlässigkeit und Sympathie eingeordnet wird und dann per Zufall eine richtige, falsche oder gar keine Nachricht erhält. EVEs werden abgespeichert und über die Gesamtheit der wahrscheinlichste Treffpunkt ermittelt. Wenn die PEVE Nachricht ausgelesen wird, so testet der Clubster ob er richtig lag und erneuert seine Verlässigkeitswerte. Außerdem werden so auch die WUW Nachrichten des Clubsters erstellt, wenn der Clubster nicht eingeweiht ist. Aus einer Abfolge von Fragearten sucht sich der Clubster immer einige Clubsters aus die dieser Frageart (z.B. zuverlässig und sympathisch) entsprechen und stellt entweder allen oder nur einer gewissen Anzahl eine Frage. Wenn eine gewissen Anzahl der so gestellten Fragen überschritten ist, wird keine weitere Frage mehr gestellt.

Erweiterung(en):

Erweitert habe ich mein Programm dadurch, dass ein Clubster der nicht von ULI informiert worden ist auch von anderen Clubsters Fragen gestellt bekommt und diesen ja vielleicht im Laufe des Tages auch antworten könnte. Dabei soll der Clubster nur antworten, wenn die Sicherheit mir der er den richtigen Treffpunkt weis einen gewissen Wert überschreitet. Bis zu diesem Zeitpunkt sollen die Anfragen zurückgehalten werden.

Programmdokumentation:

Verteilung des Quelltextes:

Im Unit `NACHRICHTEN.PAS` befinden sich die Typenbezeichnungen für die SMS Übermittlung und die Bildung von Statistiken. Ich habe sie ausgesondert, da sie in `ULI1.PAS` und `CLUBSTER1.PAS` verwendet werden.

Wird das Programm gestartet, so wir einmal das ULI Fenster geöffnet. Dessen Quelltext befindet sich in `ULI1.PAS`. Die anderen Units (`CLUBSTER1.PAS, NACHRICHTEN.PAS, DIAGRAMM1.PAS`) sind zwar auch in `AUFGABE2.DPR` vorhanden, werden aber nur aufgeführt, damit Delphi sie beim kompilieren nicht vergisst. ULI1.PAS hat beinhaltet die Prozeduren die zur Nachrichtenübermittlung und Steuerung der einzelnen Clubster nötig sind. Wichtiger dabei ist, dass `ULI1.PAS` die anderen Units verwendet.

`DIAGRAMM1.PAS` ist dabei wie bei einem Standardformular aufgebaut. Da es nur zum anzeigen eines Diagramms verwendet wird, und von sich aus überhaupt nicht macht, hat es eigentlich auf keinen ausführbaren Programmteil.

`CLUBSTER1.PAS` weicht hingegen vom Delphi Standardformular ab, da es selbst sich nicht als Variable kennt. Die Zeile `VAR clubster : Tclubster;` ist nicht vorhanden. Da ja nicht nur ein Clubster Fenster vorhanden sein soll sondern (fast) beliebig viele, wird in `ULI1.PAS` ein `ARRAY` deklariert, dessen Elementen dann Fenster zugeordnet werden (`clubsters: ARRAY OF Tclubster;`). Im CLUBSTER1.PAS Unit befinden sich die Prozeduren durch welche die einzelnen Clubster sich gegenseitig Fragen und Antworten schreiben.

Typendefinitionen

Alle Typendefinitionen (außer die der Fenster) befinden sich in NACHRICHTEN.PAS

`TYPE tart = SET OF (eve, peve, wuw);`
Für die drei verschiedenen Arten von Nachrichten die es im Kommunikationsprotokoll gibt.

`TYPE Tinhalt = RECORD art: tart; ort, zeit: byte END;`
Beinhaltet die zwei Variablen für Ort und Zeit und die Variable die den Typ der Nachricht festlegt. Ort und Zeit sind natürlich nur dann wichtig wenn es eine EVE Nachricht ist.

`TYPE nachricht = RECORD von: byte; inhalt: Tinhalt END;`
Dieser Datenrecord aus Tinhalt und der von-Variable (für Sender oder Empfänger) wird im Programm verwendet um die Nachrichten von einem Buffer zum anderen zu transportieren.

Die anderen 3 Typendefinitionen :

`TYPE Tauswertung = RECORD anfragen, falsche, richtige: byte; wahl, eingeweiht: boolean; END;` wird für die Auswertung der einzelnen Clubster verwendet. Der Clubster schreibt in die Variablen wie viele Anfragen, falsche bzw. richtige Antworten er an diesem Tag bekommen hat und ULI lest sie dann aus.

`TYPE Tclubster_auswertung = RECORD cyberin, richtige, falsche, anfragen, eingeweiht: word END;` wird verwendet um in ULI einen ARRAY `clubster_auswertung` zu erstellen, der für jeden einzelnen Clubster die Summe der Cyberin-Tage, etc. speichert.

`TYPE TULI_auswertung = RECORD traffic, cyberin, richtige, falsche, anfragen, eingeweiht: word END;` als Varablentyp für den `ULI_auswertung ARRAY` welcher den Verlauf über mehrere Tage beinhaltet.

Clubsters und deren interne Prozeduren:

Erst einmal zu den Variablen die ein Clubster mit sich führt. Veröffentlicht werden von ihm

`auswertung: Tauswertung;` erneuert der Clubster jeden Tag neu, damit ULI nach Sendung seines PEVEs nachlesen kann wie viel der einzelne Clubster gemacht hat.
`meinenummer: byte;` enthält die Nummer des Clubsters, damit er nicht an sich selbst Anfragen stellt.
`mitgliederanz: byte;` enthält die Anzahl der Clubsters. ULI verändert diese Zahl immer wenn er eine neuen Clubster erstellt oder einen löscht.
`sendbuffer: ARRAY OF nachricht;` speichert die Nachrichten die sich ein Clubster zum herausfindes des Treffpunkts zurechtgelegt hat und die Nachrichten die er als Antwort an andere Clubster schreibt. ULI holt sich aus diesem Buffer die Nachrichten.
`getbuffer: ARRAY OF nachricht;` beinhaltet alle Nachrichten die ein Clubster bekommen aber noch nicht verarbeitet hat.
`verlass: ARRAY OF integer;` speichert die Verlässigkeitswerte die sich ein Clubster in laufe der Tage über die anderen bildet.
`sympathiewerte: ARRAY OF shortint;` speichert die Sympathiewerte die er gegenüber den anderen Clubster einnimmt. –1 entspricht dabei unsympathisch, 0 neutral und 1 sympathisch.

Diese Variablen stehen im PUBLIC Bereich, da sie zur Kommunikation zwischen den ULI und den Clubstern verwendet werden. Im PRIVATE Bereich befinden sich für den Ablauf auch noch sehr wichtige Variablen:

`uli_eve: boolean;` nimmt true an wenn der Clubster von ULI informiert wurde.
`fragestunde: boolean;` wird zu Beginn eines Tages auf false gestellt und nimmt true an, wenn sich der Clubster seine Fragen an die anderen schon ausgedacht hat.
`beste_wahl: Tinhalt;` beinhaltet den besten Treffpunkt den sich der Clubster aus den Antworten berechnet hat.
`sicherheit: real;` nimmt einen Wert an der einer Sicherheit des besten Treffpunkts entspricht.
`message_book: ARRAY OF Tinhalt;` speichert für jede Person die jeweilige Fragesituation. Wenn im Inhalt eine WUW steht, dann hat der Clubster an diese Person eine Nachricht verschickt, oder wenn es ein EVE mit Ort enthält hat die Person auch geantwortet. Dieser ARRAY wird für die Bildung der Verlässigkeitswerte verwerndet.

Das Clubster Formular hat zwei Prozeduren `neuer_tag` und `bearbeiten`, welche von ULI zur Simulation aufgerufen werden.
`Neuer_tag` wird wie der Name schon sagt jeden Tag „am Morgen“ aufgerufen. Die Prozedur setzt die Variablen zurück, so das für den Clubster wieder ein neuer Tag beginnen kann. `Uli_eve` und `fragestunde` werden auf false gesetzt, er beste Treffpunkt auf 0, das Fenster erhält wieder den normalen grauen Farbton, `sendbuffer` und `getbuffer` werden entleert und das `Messagebook` wieder neutralisiert.
Die Funktion `bearbeiten` übernimmte den eigentlichen Teil der Aktionen des Clubsters. Sie wird mehrmals am Tag von Uli aufgerufen und ermöglicht somit dem Clubster zu agieren. Ihr Ablauf funktioniert so:

Nachricht aus getbuffer holen;

Nachricht ist EVE
  Nachricht ist von ULI
    uli_eve auf true; sicherheit ist 100%; beste_wahl ist Nachrichteninhalt; Fenster grün machen;
  ELSE
    Nachricht ins Messagebook schreiben; Prozedur wohin aufrufen;

```
Nachricht ist PEVE
  Beste_wahl ist falsch
    Fenster rot machen; Im Auswertungsarray wahl auf false setzen;
  ELSE
    Im Auswertungsarray wahl auf true setzen;
  Schleife für alle Mitglieder;
    Im Messagebook steht eine EVE Nachricht für diese Person
      Im Auswertungsarray anfragen erhöhen;
      Die Information von diesem Clubster war richtig
        Verlässigkeitswert um 2 erhöhen; Im Auswertungsarray richtige erhöhen;
      ELSE
        Verlässigkeitswert um 3 erniedrigen; Im Auswertungsarray falsche erhöhen;
    ELSE im Messagebook steht eine WUW Nachricht
      Im Auswertungsarray anfragen erhöhen; Verlässigkeitswert um 1 erniedrigen;

Nachricht ist WUW aber Person ist nicht von ULI informiert und Treffpunkt zu unsicher
  Nachricht wieder an das Ende des Getbuffers hängen;

Nachricht ist WUW und Treffpunkt sicher genug
  richtige Antwort erstellen;
  falsche Antwort erstellen;
  Nach Sympathiewert und Verlässigkeit Antwortstrategie aus passender textbox auslesen;
  Wenn Strategie größer 0
    mit dem Wahrscheinlichkeit der Antwortstrategie falsche oder richtige Antwort in Sendbuffer schreiben;

Nicht von ULI informiert und noch keine Fragen gestellt
  Schleife für alle Elemente im Listenfeld
    Inhalt des Listenfelds auf TYP und ANZAHL aufspalten
    WIEDERHOLE
      WIEDERHOLE
        WIEDERHOLE bis Zufallsperson nicht selbst und noch ungefragt;
        Test ob diese Person dem Fragetyp entspricht;
      BIS Fragetyp entspricht ODER über 100 Versuche;
      Wenn Fragetyp entspricht und maximale Fragenanzahl nicht erreicht
        Frage an diese Person senden;
    BIS Anzahl von diesem Fragetyp erricht ODER maximale Fragenanzahl erreicht ODER über 100 Versuche
```

Die Entscheidung welchen Treffpunkt sich der Clubster aus dem `Messagebook` auswählt übernimmt die Prozedur `wohin`. Sie wird immer dann aufgerufen wenn der Clubster aus dem `Getbuffer` eine EVE Nachricht ausliest. Außerdem ermittelt die Prozedur die Ermittlung der `Sicherheit` mit welcher der Clubster rechnen darf.
Um dies zu erreichen bildet die Prozedur einen Array mit den ganzen verschiedenen Antworten die ein Clubster erhalten hat und im Messagebook abgespeichert hat. Parallel zu diesem Array existiert ein Array mit der Häufigkeiten mir der eine Antwort aufgetreten ist. Die Antwort mit der höchsten Häufigkeit wird ausgewählt.
Um dem Clubster nicht schon bei einer Antwort eine 100%ige Sicherheit zu suggerieren, erhöhe ich den Divisor `messageanz` immer auf minimal 3. So erreiche ich, dass ein Clubster der nur eine Nachricht erhält sich auch nur 33% sicher ist.
Halbformal sieht dieser Programmabschnitt so aus:

```
Schleife für alle Mitglieder

  EVE Nachricht von dieser Person im Messagebook vorhanden
    Nachrichtenanzahl um 1 erhöhen;
    vorhanden auf false setzen;
    Schon eine Antwort im Antwortenarray
      Schleife für alle Antworten im Antwortenarray
        Wenn Antwort mit der im Antwortenarray abgespeicherten übereinstimmt
          Counter für diese Antwort um 1 erhöhen;
          vorhanden auf true setzen;
    Wenn Vorhanden auf false
      Antwort zum Antwortenarray hinzufügen
      Counter für diese Antwort auf 1 setzen;

Wenn Nachrichtenanzahl kleiner 3
  Nachrichtenanzahl auf 3 setzen;

Wahrscheinlichste Antwort 0 annehmen;
Wahrscheinlichkeit für Antwort[0] berechnen;

Schleife für alle Antworten im Antworten Array;
  Wenn Wahrscheinlichkeit kleiner Wahrscheinlichkeit für Antwort[i]
    Wahrscheinlichste Antwort i annehmen;
    Wahrscheinlichkeit für Antwort[i] berechnen;

Sicherheit ist Wahrscheinlichkeit
Wahrscheinlichster Treffpunkt ist Wahrscheinliste Antwort;
Farbe des Fensters nach Sicherheit setzen;
```

Nachrichten übermitteln

Das übermitteln der Nachrichten übernimmt das ULI Fenster, genauer gesagt die Funktion `nachrichten_uebermitteln`. Um Nachrichten von einem Clubster zum anderen Clubster zu bringen liest es aus dem `Sendbuffer` des einen Clubsters eine Nachricht aus. Die Nachricht geht an die Person mit der Nummer die sich in der Nachricht in der `von` Variable befindet. Bevor der Empfänger aber die Nachricht erhält wird die `von` Variable in der Nachricht noch auf die Nummer des Senders gesetzt. So kann der Empfänger dem Sender auch wieder ein Antwort zukommen lassen. Die Funktion führt diese Abfolge pro Clubster genau einmal aus. Wenn sie dabei aber nichts mehr zu tun hat, sprich alle `Sendbuffer` leer sind, nimmt die Funktion den Wert false an, sonst true.

Simulieren eines Tages

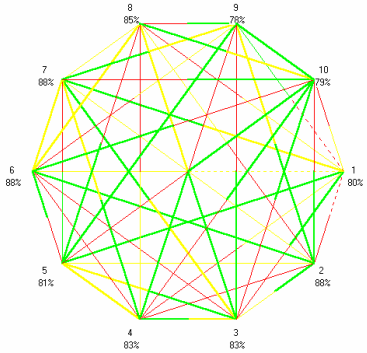
Die Funktion sim_day übernimmt im Fenster ULI die Simulation eines Tages. Der Ablauf eines Tages sieht bei dabei so aus. Zuerst werden die Prozeduren `neuer_tag` in den einzelnen Clubsters ausgeführt. Dies geschieht über die Prozedur `clubsterinit` (hab ich ausgelagert, weil ich sie beim hinzufügen der Clubsters auch brauche um sie zu initialisieren).
Dann wird über die Funktion `uli_starteve(anzahl, typ);` eine gewisse Anzahl von EVE Nachrichten von Uli versandt. Durch den Typ kann verstellt werden, ob immer gleich viele Clubsters eine Nachricht erhalten oder jedem Clubster mit einer gewissen Wahrscheinlichkeit eine Nachricht zugesandt wird.
Danach wird das erstemal die Funktion `clubsters_tippen` ausgeführt. Diese Funktion führt bei jedem Clubster einmal die Prozedur `bearbeiten` aus. Dadurch ermöglicht sie es den Clubsters zu agieren. Sie selbst nimmt den Wert True an, wenn ein Clubster wirklich noch etwas zu tippen hat, sprich noch Nachrichten in seinem `Getbuffer sind`.
Darauf folgt eine Schleife die so lange die Funktionen `nachrichten_uebermitteln` und `clubsters_tippen` ausführt, bis beide false werden, aber maximal 100 mal.
Danach versendet ULI an alle Clubster Mitglieder eine PEVE Nachricht über die Prozedur `uli_eve`. Jetzt wird die Funktion `clubsters_tippen` noch so lange ausgeführt bis sie false wird. Ist dies abgeschlossen wird die Prozedur `auswertung` ausgeführt, welche wie folgt funktioniert.

Auswertung

Die Auswertung der Simulation funktioniert so. Ich habe zwei verschiedene Arrays vom Typ Tclubster_auswertung und TULI_auswertung. Der eine Speichert für jeden Clubster wie viele Anfragen er versandt hat, wie viele falsche Antworten er darauf erhalten hat usw. Dies summiert er über mehrere Tage auf. Im zweiten Array funktioniert die Auswertung immer für jeden Tag gesondert. In ihm wird für jeden Tag gespeichert wie viele Clubsters cyberin waren, wie viele Fragen gestellt wurden usw. Um dies zu gewährleisten wird jeden Tag „am Abend“ die Prozedur `auswertung` ausgeführt. Diese Prozedur geht jeden einzelnen Clubster durch und liest seine `auswertungs` Variable aus und verändert nach den Vorgaben dieser die Werte in den ULIs Auswertungsarrays.
Über die zwei Funktionen `club_erstellenClick` und `ULI_erstellenClick` werden Textdateien erstellt, die diese Arrays mit Tabulator getrennt wiedergeben.

Das Netzwerk-Diagramm

Habe ich eingebaut um die Funktionsfähigkeit der Clubsters zu visualisieren. Es ist so zu verstehen:
Grün entspricht sympathisch, gelb neutral und rot unsympathisch. Hierbei ist zu beachten, dass eine grüne Linie beim Clubster bedeutet, dass ihn der andere als sympathisch empfindet. Dick gezeichnet werden Linien, wenn ein Clubster den anderen einen Verlässigkeitswert von über 3 zuweist und gestrichelt wenn er unter −5 gesunken ist.



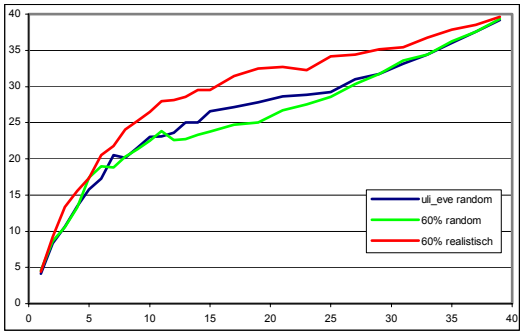
Probleme:

Das erste Problem entsteht dadurch, dass jeder Clubster ein eigenes Fenster mit vielen Bedienelementen hat. Beim Programmieren hatte ich zum Testen immer nur 20 Fenster offen. Als ich dann aber eine Simulation mit 100 Clubsters starten wollte, wurde ich mit einem kompletten Systemabsturz bestraft als ich meinem ersten Clubster eine Konfiguration geben wollte. Scheinbar überlasten die vielen Clubsters meine Maschine zu stark, denn an meiner Zivildienststelle konnte ich weitausmehr erstellen.

Das nächste Problem ist eigentlich gar kein Problem sondern nur eine Ungenauigkeit bei der Simulation. Bei meiner Simulation kommen nie zu einem echten Treffpunkt. Dadurch entsehen natürlich Ungenauigkeiten. Nehmen wir ein gar nicht mal so unwahrscheinliches Beispiel. Zwei Freunde erhalten beide kein EVE von Uli und beide fragen sich natürlich gegenseitig. Einer von beiden erfährt nach einiger Zeit einen falschen Treffpunkt und schickt auch eine Antwort an seinen Kumpel. Dadurch treffen sich beide abends an einem falschen Treffpunkt. In meiner Simulation hat es zur Folge, dass die Verlässigkeitswerte der beiden sinken, da der eine ja nicht in Betracht zieht, dass die Antwort die er erhalten hat ja nach besten Wissen und Gewissen war. Diese Situation entsteht mitunter auch wegen der Erweiterung, dass nicht eingeweihte Clubsters auch antworten, und der Fall kann vermieden werden indem man die Sicherheit die ein Clubster braucht um EVEs zu versenden größer als 33% ist.

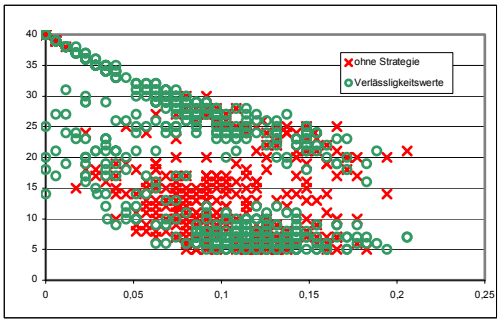
Fragestellungen:

Gibt es einen Zusammenhang zwischen den prozentualen Anteil jener, die morgens ein Event-SMS erhalten und jener, die Abends cyber-in erschienen?



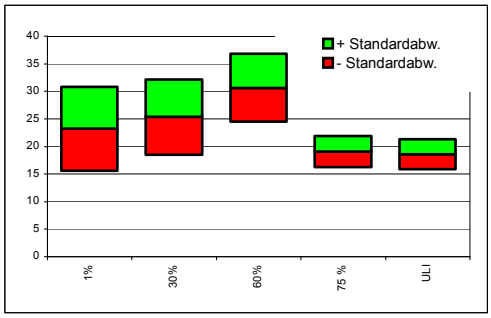
Es ist ganz klar das es da einen Zusammenhang geben muss, da wenn keiner eine SMS erhält 0 Personen cyberin sind und bei 100%iger Benachrichtigung alle kommen (bis auf die, die keine Lust haben, und nachts lieber programmieren). Die Frage ist nur, wie schaut es im Übergang aus.
Um dies festzustellen habe ich 40 Clubsters die per Zufall die vorgegebenen Fragestrategien verwenden über jeweils 20 Tage simuliert. Dabei habe ich die Antwortstrategie zwischen „Clubster antwortet nur wenn er von ULI benachrichtigt worden ist“ auf „Anwortet ab 60% Sicherheit“, bzw. die Sympathiewerteverteilung einmal random das andere mal realistisch.
Auffällig dabei ist, das die Anzahl der cyberin Personen anfangs bei jeder Variante ähnlich sind, und sich ab 6 EVE Nachrichten von Uli trennen.
Da die entstandenen Kurven zu Beginn sehr stark ansteigen und dann flacher werden könnte man sie durch eine Wurzelfunktion annähern.

Gibt es einen Zusammenhang zwischen dem Prozentsatz falsch beantworteter SMS-Anfragen und der Anzahl jener, die abends cyber-in erscheinen?



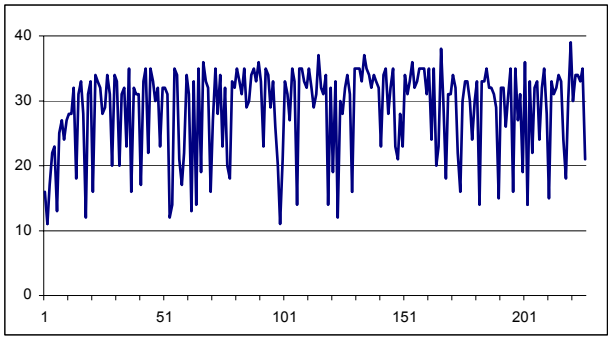
Um diese Frage zu beantworten habe ich wieder 40 Clubsters simuliert. Dabei verwendeten alle die gleiche Fragestrategie „verlässliche dann alle“ oder „alle“ was eigentilch keiner Strategie entspricht. ULI hat immer 5 Clubster informiert und ich die Wahrscheinlichkeit mit der die Anfragen richtig beantwortet werden mehrmals variiert.
Im Diagramm kann man deutlich erkennen das die Anzahl der Personen die Cyberin sind mit steigendem Anteil der falschen Antworten sinkt.
Bei der Variante bei denen die Clubsters ohne Strategie befragen, kommen durchschnittlich weniger, als bei denen mit Strategie. Außerdem erreichen nicht eine so niedrige Anzahl von falsche Antworten, da sie ja einfach irgendjemand Fragen.
Bei der Betrachtung dieses Diagramms ist mit noch etwas aufgefallen, woraus ich mir weitere Fragen gebildet habe.

Wie hängt die Sicherheit, welche Clubsters brauchen um Nachrichten zu beantworten, mit dem Auftreten einer kettenreaktionsähnlichen Informationsverteilung zusammen?



Bei gewissen Frage und Antwortkonfigurationen kommt es bei mehreren aufeinanderfolgenden Simulationen vor, dass einmal nur wenige Clubster vom Treffpunkt Bescheid wissen und dann wieder fast alle. Eine dieser Konfigurationen ist:
Antwortstrategie: eigenes Beispiel; Fragestrategie: Verlässliche dann Alle, mit maximal 5 Fragen und 4 Benachrichtigungen von ULI, alle Clubster haben ausgeprägt Verlässigkeitswerte, 50 Simulationstage pro Sicherheitsvariante
Ich konnte keine besseren Messwerte liefern, da ich leider nicht ausreichend viele Clubster in verwenden konnte. Dennoch ist zu erkennen, dass es irgendwo ein Maximum geben muss. Bei mir ist es gerade bei 60% was genauer gesagt bedeutet, dass ein Clubster wenn er zwei gleiche Nachrichten erhalten hat diese Informationen an alle anderen weitergebt.
Die Unterschiedlichen Standardabweichungen kann man so deuten: Wenn die Clubster auch schon bei unsicherem Informationsstand den Treffpunkt weitergeben, so ist es zwar sehr wahrscheinlich, dass alle Clubster informiert sind. Wenn aber am Anfang der Informationskette eine Fehlnachricht eingeschleust wurde so, kann es leicht vorkommen das sich viele an einem falschen Ort treffen. Ähnlich wahrscheinlich ist aber auch, dass alle Clubster ehrlich waren und dadurch sich fast alle am selben Ort treffen. In diesen Fällen kann man von einer kettenreaktionsähnlichen Informationsverbreitung sprechen.
Müssen sich aber die Clubster erst ziemlich sicher sein um den Treffpunkt weiterzugeben, tritt nie eine Kettenreaktion ein. Dadurch beleibt die Standardabweichung auch so gering.

Wie wirkt sich die Erfahrenheit der Clubsters auf die Anzahl der Cyperin Clubsters aus?
Um dies zu ermitteln, erhielten alle Clubster die Fragestrategie 2 aus der Angabe. Die Sympathiewerte habe ich auf zufällig gesetzt und eine Simulation über mehrere Tage gestartet. Wenn ein Clubster sich noch keine Verlässigkeitswerte gebildet hat, muss er wohl oder übel x beliebige Personen fragen. Das hat zur Folge dass er nicht nur ihm gegenüber sympathisch gesonnene Clubster erwischt sonder auch mal welche die ihm eins auswischen wollen. Aus diesem Grund sind am Anfang der Simulation die cyberin-Werte relativ gering, nehmen aber dann schnell zu und pegeln sich aber dann auf einem relativ Konstanten Maximum aus mit ein paar Ausbrüchen nach unten.



Strategien:

Welche Strategien sind besonders erfolgfrei?
Ein unwissender Clubster möchte möglichst oft Cyberin sein, ein wissender möglichst seine Freunde aber nicht seine ungeliebten Bekannten treffen.
Ein paar Fragestrategien zur Auswahl:

1. Clubster fragt nur ihm sympathische
2. Clubster fragt Verlässliche und den Rest auf 5 Fragen irgendjemanden.
3. Clubster befragt maximal 4 verlässliche und den Rest auf 5 Fragen irgendjemanden.

	Realistische Sympathiewerte	Random Sympathiewerte
Strategie1	95,2%	77%
Strategie2	40%	70,4%

Strategie Nr.1 ist unschlagbar, wenn man davon ausgeht das Sympathie irgendwie auf Gegenseitigkeit beruht. Wenn ein Clubster jemanden als sympathisch einstuft, würde ich im normalen Leben davon ausgehen, dass ihn der andere auch leiden kann und ihn deshalb zu 100% eine richtige Nachricht zusendet. Wenn man alle Clubster so Konfiguriert und die Sicherheit noch auf 1% setzt(falsche Antworten sollte es ja nicht geben) ist es eine dieser Fälle in denen eine Simulation nur noch wenig ergiebig ist. Wenn man nicht die Situation durch möglichst viele Clubster und möglichst wenige Benachrichtigungen von ULI ins extreme treibt, so erhalten alle Clubster über mehr oder weniger Zwischenstufen eine Nachricht des Treffpunktes.
Da durch die realitätsnähere Vergabe von Sympathiewerten das ganze Simulieren nicht so lustig ist, vergebe ich deshalb immer nur die Sympathiewerte per Zufall.
Dadurch verliert Strategie Nr.1 ihre Qualität. Ein Clubster nach diesem System vorgeht, fragt immer Clubster die er Sympathisch findet, ihn aber womöglich die ganze Zeit an der Nase herumführen und er merkt es nicht einmal (Es ja eigentlich eine Gemeinheit einem Clubster diesen Befehl zur Dummheit zu geben).
Strategie Nr.2 und Nr.3 sind da besser. Sie befragen zwar Anfangs Kreuz und Quer, lernen aber relativ schnell zuverlässige Clubster kennen, die sie dann täglich mit Fragen löchern können. Stellt sich nur noch die Frage welche unter diesen beiden die bessere Strategie ist.
Ein Clubster der nach Strategie 2 arbeitet, funktioniert meistens so: Er befragt am Anfang seines Lebens einige Clubster, bis er 5 gefunden hat die er dann immer fragt, weil es die einzigen sind mit ausriechend hohem Verlässigkeitswert und dieser wird dann auch noch immer höher. Meistens deshalb weil ja der Fall eintreten kann, das die `tries` Variable > 100 wird obwohl er schon 5 Verlässliche kennt.
Ein Clubster der nach der Strategie 3 vorgeht hat ein etwas abwechslungsreicheres Leben. Er befragt immer eine Person per Zufall und für die restliche 4 pickt er sich schon verlässliche Clubster heraus.

Strategie 1	76,2%
Strategie 2	43,2%

Persönlich würde ich jedem Clubster die Strategie 3 raten, da er wenn er schon hauptsächlich mit SMS kommuniziert jeden Tag jemanden neuen auf die Verlässigkeit testen kann. Allerdings muss er sich dann auf starke Cyberin Einbussein einstellen. Bei einer Simulation bei der gleich viele Personen die gleiche Strategie verwendet haben haben sich Folgende Werte ergeben:

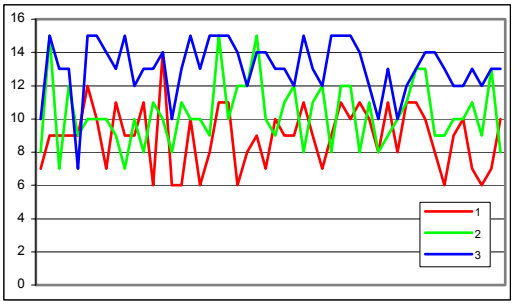
	Cyberin	Eingeweiht	Richtige Antworten	Richtig beantwortet
Strategie 1	32%	7%	61%	11%
Strategie 2	52%	8%	95%	18%
Strategie 3	36%	7%	68%	13%

Man sieht das die Tests untereinander auch im Zustammenspiel aller drei übereinstimmen.

Ausserdem habe ich auch noch 3 verschiedene Fragestragegien getestet:

1. Clubster antworten sympathischen Mitgliedern korrekt und den anderen gar nicht.
2. Clubster antworten sympathischen Mitgliedern korrekt, neutralen 50% richtig und den unsympathischen 100% falsch.
3. Clubster unterscheidet nicht nur nach Sympathiewerten sondern auch nach Verlässligkeit, die stäreke Gewichtung liegt auf den Sympathiewerten. Clubster antwortet bei 60% Sicherheit.

Dabei viel es mir allerdings sehr schwer irgendwie einzuordnen welche dieser Strategien die bessere ist. Man könnte zwar sagen, dass die beste Strategie diejenige sei, bei der sich nur wenige Clubster treffen, weil dann nur Leute dabei sind, die man sympathisch findet. Dann würde eine Ornung so aussehen: Strategie 1 (44,8%), Stragegie 2 (51,5 %), Strategie 3 (65,5%) Dabei ist aber zu beachten das die 60% Sicherheit bei der Strategie 3 eine Steigerung der Cyberin-Werte mit sich führt, die aber für ein einzelnen Clubster und seine Freunde wahrscheinlich eher positiv auswirken. Das kann ich aber nicht genau testen, da mein Auswertungsystem nicht so mächtig ist. Dazu wären echte Treffen nötig, bei denen die einzelnen Clubster dann einen Qualitätswert für das Treffen bestimmten.



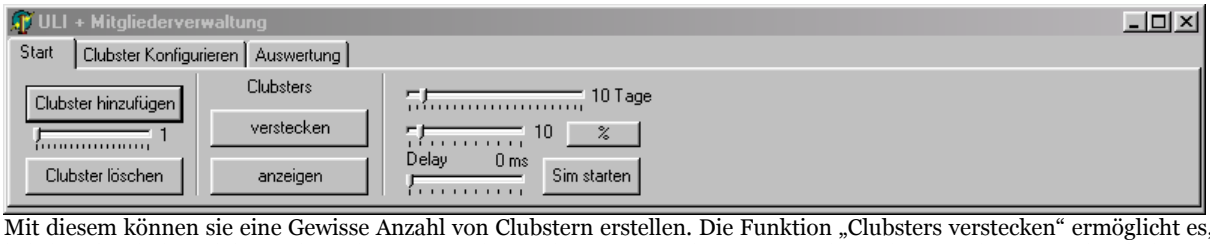
Extreme Strategien:

Natürlich gibt es extreme Strategien die eine Simulation uninteressant machen. Die einfachsten Beispiele hierfür sind:

- Nur ein Clubster existiert
- Zu wenig oder zu viele EVEs die ULI versendet
- Sendestrategien bei denen nur falsche Antworten gegeben werden
- Fragestrategien bei denen 100% der anderen Clubstern eine Frage gestellt wird.

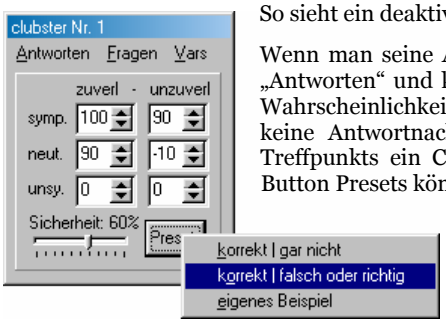
Programmablaufprotokoll:

Beim Start des Programms erscheint nur das ULI Fenster.

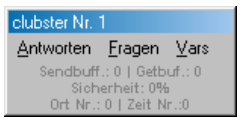


Mit diesem können sie eine Gewisse Anzahl von Clubstern erstellen. Die Funktion „Clubsters verstecken“ ermöglicht es, bei der Simulation nicht Rechenzeit wegen andauernden Screenupdates zu verschwenden. Vor längeren Simulationen, sollte man die Clubsters unbedingt ausblenden, da man sonst eine paar Minuten auf des Ergebnis warten muss. Rechts befinden sich Schieberegler mit denen man bestimmen kann über wie viele Tage simuliert werden soll, wie viele Personen informiert werden und wie lange bis zum nächsten Simulationstag pausiert werden soll. Der Knopf „Sim starten“ startet die ganze Simulation.

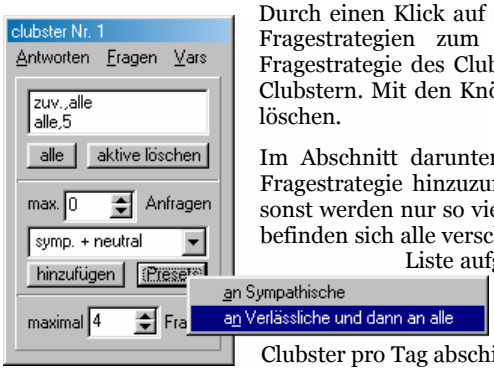
Aber bevor man eine Simulation startet, sollte man seinen Clubstern eine Strategie geben.



So sieht ein deaktivierter Clubster aus.



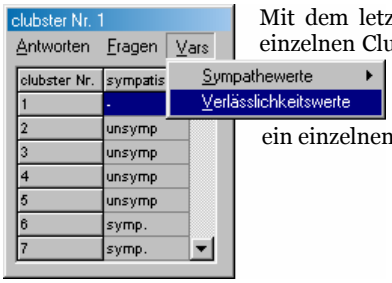
Wenn man seine Antwortstrategie verändern will, so klickt man auf „Antworten“ und kann dann für jeden Typ von Person die passende Wahrscheinlichkeit für eine richtige Antwort einstellen. -10 bedeutet keine Antwortnachricht schicken. Der Schieberegler stellt ein ab welcher Sicherheit des Treffpunkts ein Clubster antwortet, obwohl er nicht von ULI informiert wurde. Über den Button Presets können die von mir benutzten Beispiele direkt abgerufen werden.



Durch einen Klick auf den „Fragen“ Knopf kommen die Einstellmöglichkeiten für verschiede Fragestrategien zum Vorschein. Die oberste Liste zeigt die momentan eingestellte Fragestrategie des Clubsters. Er befragt zuerst alle zuverlässigen und dann noch 5 von alles Clubstern. Mit den Knöpfen alle oder aktive löschen kann man Einträge wieder aus der Liste löschen.

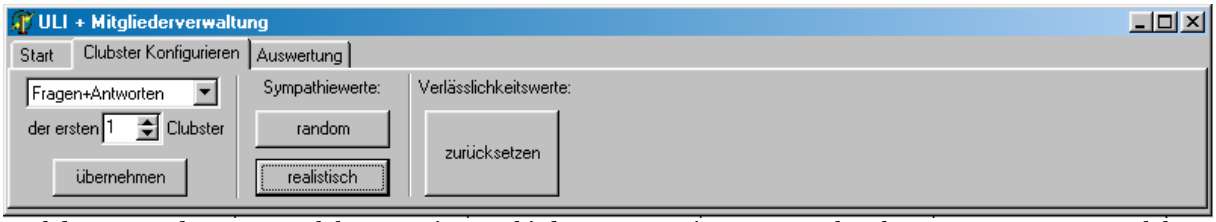
Im Abschnitt darunter befindet sich der Bereich um die einzelnen Personengruppen zur Fragestrategie hinzuzufügen. Wenn maximale Anfragen auf 0 gestellt ist bedeutet dies alle, sonst werden nur so viele Clubsters aus dieser Personengruppe befragt. Im Dropdown Fenster befinden sich alle verschiedenen Möglichkeiten, die mit dem „hinzufügen“ - Knopf in die obere Liste aufgenommen werden. Durch einen Druck auf Presets kann man auch hier direkt die verwendeten Beispiele aktivieren.

Im Feld maximale Fragen kann man einstellen wie viele Fragen der Clubster pro Tag abschickt.



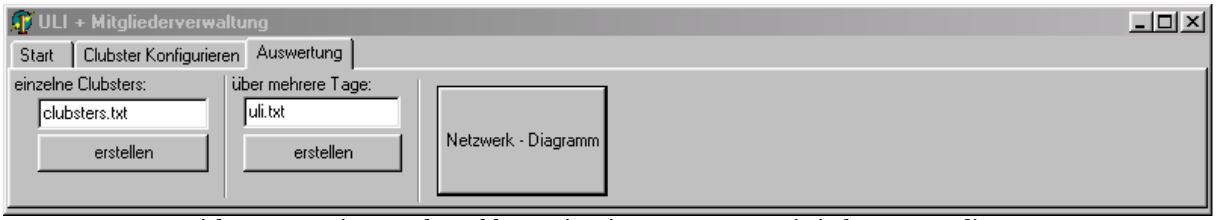
Mit dem letzte Menüpunkt, kann man die Sympathiewerte und die Verlässigkeitswerte der einzelnen Clubster einsehen. Außerdem können unter Sympathiewerte→random die Werte für diese Peson zufällig gesetzt werden.

Sind in der Liste gerade die Sympathiewerte angezeigt, so kann nur einen Klick ein einzelnen Wert geändert werden.



Nachdem man den ersten Clubstern mit verschiedenen Strategien versorgt hat kann man nun unter Clubster Konfigurieren, diese auf alle anderen Clubsters per Zufall verteilen. Dabei kann man angeben ob nur der erste oder die ersten 2, 3, ... als Beispiel verwendet werden sollen.

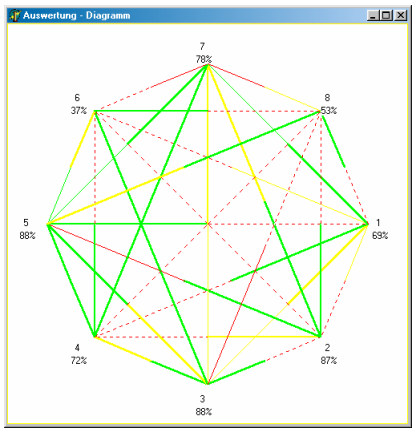
Außerdem kann man hier die Sympathiewerte aller Clubster festsetzen. Realistisch bedeutet, dass Sympathie und Antipathie zweier Personen zueinander auf beiden Seiten gleich sind. Des weiteren können die Verlässlichkeitswerte aller Clubster auf 0 zurückgesetzt werden.



Das Programm speichert wenn ein Tag abgeschlossen ist eine Menge an statistischen Daten dieses Tages. Wenn man diese Daten für weitere Bearbeitung benötigt, kann man sich 2 verschiedene Tabellen im Textformat ausgeben lassen. So sieht zum Beispiel die Tabelle für 8 verschiedene Clubsters für ein paar Tage aus.

ClubsterNr.	cyberin	eingeweiht	richtig	falsch	Anfragen
1	47	22	91	18	184
2	59	16	88	8	208
3	60	20	100	5	192
4	49	18	94	21	200
5	60	19	95	6	196
6	25	13	95	38	220
7	53	11	142	7	228
8	36	17	68	54	204

Ähnlich aussagekräftig ist auch das Neztwerkdiagramm, welches aber ab einer gewissen Anzahl von Clubstern etwas zur Unübersichtlichkeit tendiert.



Quelltexte:

Nachrichten.pas:

```
UNIT nachrichten;

INTERFACE
TYPE tart = SET OF (eve, peve, wuw);
TYPE Tinhalt = RECORD art: tart; ort, zeit: byte END;   //Inhalt einer Nachricht
TYPE nachricht = RECORD von: byte; inhalt: Tinhalt END;         //Eine Nachricht

//Typen zur Bildung von Statistiken
TYPE Tauswertung = RECORD anfragen, falsche, richtige: byte; wahl, eingeweiht: boolean; END;
TYPE Tclubster_auswertung = RECORD cyberin, richtige, falsche, anfragen, eingeweiht: word END;
TYPE TULI_auswertung = RECORD traffic, cyberin, richtige, falsche, anfragen, eingeweiht: word END;

CONST uli_height = 140;               //Höhe des Fensters
IMPLEMENTATION

END.
```

Uli1.pas

```
UNIT ULI1;

INTERFACE

USES
  StdCtrls, Classes, Controls, Windows, Messages, SysUtils, Graphics,
  Forms, Dialogs, clubster1, nachrichten,
  Menus, ComCtrls, ToolWin, Buttons, ExtCtrls, ImgList, Spin, OleCtrls,
  vcfi, TeeProcs, TeEngine, diagramm1;

TYPE
  TULI = CLASS(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    add_clubster: TButton;
    kill_clubster: TButton;
    add_anz_track: TTrackBar;
    add_anz: TLabel;
    simulation_start: TButton;
    Bevel1: TBevel;
    Bevel2: TBevel;
    symp_rand: TButton;
    symp_rel: TButton;
    StaticText6: TStaticText;
    Bevel3: TBevel;
    clubsters_show: TButton;
    clubsters_hide: TButton;
    StaticText7: TStaticText;
    Bevel4: TBevel;
    clubsters_reset: TButton;
    eve_track: TTrackBar;
    day_track: TTrackBar;
    day_text: TLabel;
    eve_kind: TButton;
    eve_kind_popup: TPopupMenu;
    eve_perc: TMenuItem;
    eve_abs: TMenuItem;
    eve_rand: TMenuItem;
    eve_label: TLabel;
    delay_track: TTrackBar;
    StaticText1: TStaticText;
    delay_label: TLabel;
    fortschritt: TProgressBar;
    StaticText8: TStaticText;
    StaticText9: TStaticText;
    Bevel5: TBevel;
    StaticText10: TStaticText;
    verstrickung: TButton;
    Bevel6: TBevel;
    uebernehmen_art: TComboBox;
    StaticText12: TStaticText;
    uebernehmen_zahl: TSpinEdit;
    uebernehmen: TButton;
    StaticText2: TStaticText;
    club_file: TEdit;
    uli_file: TEdit;
    club_erstellen: TButton;
    uli_erstellen: TButton;
    PROCEDURE FormCreate(Sender: TObject);
    PROCEDURE simulation_startClick(Sender: TObject);
    PROCEDURE add_anz_trackChange(Sender: TObject);
    PROCEDURE add_clubsterClick(Sender: TObject);
    PROCEDURE kill_clubsterClick(Sender: TObject);
    PROCEDURE clubsters_hideClick(Sender: TObject);
    PROCEDURE clubsters_showClick(Sender: TObject);
    PROCEDURE day_trackChange(Sender: TObject);
    PROCEDURE eve_kindClick(Sender: TObject);
    PROCEDURE eve_percClick(Sender: TObject);
    PROCEDURE eve_randClick(Sender: TObject);
    PROCEDURE eve_absClick(Sender: TObject);
    PROCEDURE eve_trackChange(Sender: TObject);
    PROCEDURE delay_trackChange(Sender: TObject);
    PROCEDURE symp_randClick(Sender: TObject);
    PROCEDURE symp_relClick(Sender: TObject);
    PROCEDURE clubsters_resetClick(Sender: TObject);
    PROCEDURE verstrickungClick(Sender: TObject);
```

```
    PROCEDURE uebernehmenClick(Sender: TObject);
    PROCEDURE club_erstellenClick(Sender: TObject);
    PROCEDURE uli_erstellenClick(Sender: TObject);

  PRIVATE
    { Private-Deklarationen }
  PUBLIC
    { Public-Deklarationen }
  END;

VAR
  ULI: TULI;                                                                    //das ULI Fenster
  clubsters: ARRAY OF Tclubster;                                  //dynamischer ARRAY mit den Clubster-Fenstern
  clubsteranz: BYTE = 0;                                                        //Anzahl der Clubseter-Fenster
  ulis_eve: nachricht;                                  //Der Treffpunkt den sich ULI für diesen Tag ausgesucht hat
  clubster_auswertung: ARRAY OF Tclubster_auswertung;     //ARRAY für die Auswertung der einzelnen Clubster
  uli_auswertung: ARRAY OF TULI_auswertung;                 //ARRAY für die Auswertung der einzelnen Tage

IMPLEMENTATION

{$R *.DFM}

PROCEDURE clubsterinit;                     //Setzt alle Clubster auf die standard Werte für einen neuen
Tag
VAR
  i: BYTE;
BEGIN
  FOR i := 0 TO clubsteranz - 1 DO BEGIN
    clubsters[i].mitgliederanz := clubsteranz;
    clubsters[i].neuer_tag;
  END;
END; {clubsterinit}

PROCEDURE addclubster;            //erstellt einen neues Clubster Fenster, zeigt es an und initialisiert es
BEGIN
  inc(clubsteranz);
  setlength(clubsters, clubsteranz);

  application.CreateForm(TClubster, clubsters[clubsteranz - 1]);
  WITH clubsters[clubsteranz - 1] DO BEGIN
    top := uli_height + ((clubsteranz - 1) DIV (screen.Width DIV Width)) * Height;
    left := ((clubsteranz - 1) MOD (screen.Width DIV Width)) * Width;
    Caption := 'clubster Nr. ' + IntToStr(clubsteranz);
    meinenummer := clubsteranz;
    Visible := True;
    neuzeichnen;
  END;
  clubsterinit;
END; {addclubster}

PROCEDURE killclubster;                                                 //Löscht den letzten Clubster wieder
BEGIN
  clubsters[clubsteranz - 1].Destroy;
  dec(clubsteranz);
  setlength(clubsters, clubsteranz);
END;  {killclubster}

FUNCTION nachrichten_uebermitteln: BOOLEAN; //liest für jeden Clubster das erste Element aus dem Sendbuffer
VAR                     //und giebt es an die richtige Position weiter, dabei wird der von Wert gewechselt
  i, k: BYTE;                               //Funktion wird false wenn keine Übermittlung vorgenommen wurde
  momentane_nachricht: nachricht;
BEGIN
  nachrichten_uebermitteln := False;
  FOR i := 0 TO clubsteranz - 1 DO
    IF length(clubsters[i].sendbuffer) > 0 THEN BEGIN
      nachrichten_uebermitteln := True;
      momentane_nachricht := clubsters[i].sendbuffer[0];
      FOR k := 1 TO length(clubsters[i].sendbuffer) - 1 DO
        clubsters[i].sendbuffer[k - 1] := clubsters[i].sendbuffer[k];
      setlength(clubsters[i].sendbuffer, length(clubsters[i].sendbuffer) - 1);
      k := momentane_nachricht.von;
      momentane_nachricht.von := i;
      setlength(clubsters[k].getbuffer, length(clubsters[k].getbuffer) + 1);
      clubsters[k].getbuffer[length(clubsters[k].getbuffer) - 1] := momentane_nachricht;
    END;
END; {nachrichten_uebermitteln}

PROCEDURE uli_starteve(anzahl: BYTE; typ: STRING);                  //ULI versendet die richtigen EVE Werte
VAR
  i, nr: BYTE;
  benachrichtigt: ARRAY[0..254] OF BOOLEAN;
BEGIN
  ulis_eve.von := 255;
  ulis_eve.inhalt.ort := random(5) + 1;
  ulis_eve.inhalt.zeit := random(5) + 1;
  ulis_eve.inhalt.art := [eve];                     //Der neue Treffpunkt wird ein eine EVE Nachricht gepackt

  FOR nr := 0 TO clubsteranz - 1 DO BEGIN
    benachrichtigt[nr] := False;
    setlength(clubsters[nr].getbuffer, 0);
  END;
  IF typ = '%' THEN BEGIN
    typ := 'absolut';
    anzahl := round(clubsteranz * (anzahl / 100))
  END;
  IF typ = 'absolut' THEN                   //Per Zufall werden anzahl Clubster mit EVE Nachrichten beglückt
    FOR i := 1 TO anzahl DO BEGIN
      REPEAT
        nr := random(clubsteranz)
      UNTIL NOT (benachrichtigt[nr]);
      benachrichtigt[nr] := True;
      setlength(clubsters[nr].getbuffer, 1);
```

```
      clubsters[nr].getbuffer[0] := ulis_eve;
      clubsters[nr].neuzeichnen;
    END ELSE IF typ = '% + rand' THEN         //Typ 3: für jeden Clubster wird entschieden ob er EVE erhält
    FOR i := 0 TO clubsteranz - 1 DO
      IF random * 100 < anzahl THEN BEGIN
        setlength(clubsters[i].getbuffer, 1);
        clubsters[i].getbuffer[0] := ulis_eve;
        clubsters[i].neuzeichnen;
      END;
END; {uli_starteve}

PROCEDURE uli_peve;                                    //ULI informiert alle Clubster wo das Treffen war
VAR
  i: BYTE;
BEGIN
  ulis_eve.inhalt.art := [peve];
  FOR i := 0 TO clubsteranz - 1 DO BEGIN
    setlength(clubsters[i].getbuffer, length(clubsters[i].getbuffer) + 1);
    clubsters[i].getbuffer[length(clubsters[i].getbuffer) - 1] := ulis_eve;
    clubsters[i].neuzeichnen;
  END;
END; {uli_peve}

FUNCTION clubsters_tippen: BOOLEAN;              //führt die bearbeiten Funktion der einzelnen Clubster aus
VAR                                                 //wird false wenn alle Getbuffer auf Null gefallen sind
  i: BYTE;
BEGIN
  clubsters_tippen := False;
  FOR i := 0 TO clubsteranz - 1 DO BEGIN
    clubsters[i].bearbeiten;
    IF length(clubsters[i].getbuffer) <> 0 THEN clubsters_tippen := True;
  END;
END; {clubsters_tippen}

PROCEDURE auswertung;                   //fügt die einzelnen Werte der auswertungs Varaibel der Clubster zu
VAR                                                          //zwei verschiedenen Auswertungstypen zusammen
  i, tagnr: WORD;
BEGIN
  IF length(clubster_auswertung) <> clubsteranz THEN BEGIN  //wird der ARRAY das erste mal benützt wird der
    setlength(clubster_auswertung, clubsteranz);   //clubster_auswertungs ARRAY auf die anzahl der Clubster
    setlength(uli_auswertung, 1);                                                          //vergrössert
    tagnr := 0;
  END ELSE BEGIN
    tagnr := length(uli_auswertung);
    setlength(uli_auswertung, tagnr + 1);
  END;

  ULI_auswertung[tagnr].traffic := 0;                              //Werte der Tages Statistik auf 0 setzen
  ULI_auswertung[tagnr].cyberin := 0;
  ULI_auswertung[tagnr].richtige := 0;
  ULI_auswertung[tagnr].falsche := 0;
  ULI_auswertung[tagnr].anfragen := 0;
  ULI_auswertung[tagnr].eingeweiht := 0;

  FOR i := 0 TO clubsteranz - 1 DO BEGIN                                 //geht die einzelnen Clubster durch
    IF clubsters[i].auswertung.wahl THEN BEGIN
      inc(clubster_auswertung[i].cyberin);
      inc(uli_auswertung[tagnr].cyberin);
    END;

    IF clubsters[i].auswertung.eingeweiht THEN BEGIN
      inc(clubster_auswertung[i].eingeweiht);
      inc(uli_auswertung[tagnr].eingeweiht);
      inc(ULI_auswertung[tagnr].traffic);
    END;

    inc(clubster_auswertung[i].richtige, clubsters[i].auswertung.richtige);
    inc(uli_auswertung[tagnr].richtige, clubsters[i].auswertung.richtige);

    inc(clubster_auswertung[i].falsche, clubsters[i].auswertung.falsche);
    inc(uli_auswertung[tagnr].falsche, clubsters[i].auswertung.falsche);

    inc(clubster_auswertung[i].anfragen, clubsters[i].auswertung.anfragen);
    inc(uli_auswertung[tagnr].anfragen, clubsters[i].auswertung.anfragen);
  END;

  inc(ULI_auswertung[tagnr].traffic, uli_auswertung[tagnr].anfragen +
    uli_auswertung[tagnr].falsche + uli_auswertung[tagnr].richtige);
END; {auswertung}

PROCEDURE sim_day;                                     //führt die simulation einen einzelnen Tages durch
VAR
  i, tries: BYTE;
BEGIN
  clubsterinit;                                                          //initalisieren der Clubster
  uli_starteve(uli.eve_track.Position, uli.eve_kind.Caption);           //EVE Nachrichten verschicken
  tries := 0;

  clubsters_tippen;
{$B+}
  REPEAT
    inc(tries);                                  //Übermittelt Nachrichten und lässte die Clubster aggieren
    sleep(uli.delay_track.Position);
  UNTIL NOT (nachrichten_uebermitteln OR clubsters_tippen) OR (tries > 100);
{$B-}

  uli_peve;                                                      //Verschickt die PEVE Nachricht an alle

  tries := 0;

  REPEAT                        //Lässt die Clubster aggieren so dass sie die EVE Nachricht im Buffer finden
    inc(tries);
```

```
  UNTIL NOT (clubsters_tippen) OR (tries > 10);

  auswertung;                                                    //Führt die Auswertung dieses Tages durch
  sleep(uli.delay_track.Position);
END; {sim_day}

PROCEDURE TULI.FormCreate(Sender: TObject);       //gibt Fenster und PageControl Objekt die richtige Grösse
BEGIN
  uli.Width := screen.Width;
  uli.Height := uli_height;
  PageControl1.Width := clientwidth;
END; {TULI.FormCreate}

PROCEDURE TULI.simulation_startClick(Sender: TObject);             //Führt eine Symulation für x Tage durch
VAR
  i: BYTE;
BEGIN
  uli_erstellen.Enabled := True;
  club_erstellen.Enabled := True;
  fortschritt.Width := day_track.Position;
  fortschritt.Max := day_track.Position;
  fortschritt.Visible := True;
  FOR i := 1 TO day_track.Position DO BEGIN
    sim_day;
    fortschritt.Position := i;
  END;
  fortschritt.Visible := False;
END; {simulation_startClick}

PROCEDURE TULI.add_anz_trackChange(Sender: TObject);     //schreibt in das Label add_anz den richtigen Text
BEGIN
  add_anz.Caption := IntToStr(add_anz_track.Position);
END; {TULI.add_anz_trackChange}

PROCEDURE TULI.add_clubsterClick(Sender: TObject);                  //erstellt x Clubster und zeigt sie an
VAR
  i: BYTE;
BEGIN
  clubsters_reset.Enabled := True;
  symp_rand.Enabled := True;
  symp_rel.Enabled := True;
  uebernehmen.Enabled := True;
  verstrickung.Enabled := True;
  clubsters_hide.Enabled := True;
  clubsters_show.Enabled := True;
  simulation_start.Enabled := True;
  FOR i := 1 TO add_anz_track.Position DO addclubster;
  uebernehmen_zahl.MaxValue := clubsteranz;
  IF eve_kind.Caption = 'absolut' THEN eve_track.Max := clubsteranz;
END; {TULI.add_clubsterClick}

PROCEDURE TULI.kill_clubsterClick(Sender: TObject);                                  //löscht x Clubster
VAR
  i: BYTE;
BEGIN
  FOR i := 1 TO add_anz_track.Position DO killclubster;
  IF eve_kind.Caption = 'absolut' THEN eve_track.Max := clubsteranz;
  IF clubsteranz = 0 THEN BEGIN
    clubsters_reset.Enabled := False;
    symp_rand.Enabled := False;
    symp_rel.Enabled := False;
    uebernehmen.Enabled := False;
    verstrickung.Enabled := False;
    clubsters_hide.Enabled := False;
    clubsters_show.Enabled := False;
    simulation_start.Enabled := False;
  END;
END; {TULI.kill_clubsterClick}

PROCEDURE TULI.clubsters_hideClick(Sender: TObject);                              //versteckt alle Clubster
VAR
  i: BYTE;
BEGIN
  FOR i := 0 TO clubsteranz - 1 DO clubsters[i].Hide;
END; {TULI.clubsters_hideClick}

PROCEDURE TULI.clubsters_showClick(Sender: TObject);
VAR
  i: BYTE;
BEGIN
  FOR i := 0 TO clubsteranz - 1 DO BEGIN                                           //zeigt alle Clubster an
    clubsters[i].Show;
    clubsters[i].neuzeichnen;
  END;
END; {TULI.clubsters_showClick}

PROCEDURE TULI.day_trackChange(Sender: TObject);        //Schieberegler für die Anzahl der simulierten Tage
BEGIN
  IF day_track.Position <> 1 THEN
    day_text.Caption := IntToStr(day_track.Position) + ' Tage'
  ELSE
    day_text.Caption := '1 Tag';
END; {TULI.day_trackChange}

PROCEDURE TULI.eve_kindClick(Sender: TObject);     //öffnet Dropdownmenu für die Art der ULI_EVE Verteilung
BEGIN
  eve_kind_popup.Popup(mouse.CursorPos.x, mouse.CursorPos.y);
END; {TULI.eve_kindClick}

PROCEDURE TULI.eve_percClick(Sender: TObject);                          //im Dropdwownmenu wird % ausgewählt
BEGIN
  eve_kind.Caption := '%';
```

```
  eve_track.min := 0;
  eve_track.Max := 100;
  eve_track.Frequency := 10;
END; {TULI.eve_percClick}

PROCEDURE TULI.eve_randClick(Sender: TObject);                    //im Dropdwownmenu wird %+rand ausgewählt
BEGIN
  eve_kind.Caption := '% + rand';
  eve_track.min := 0;
  eve_track.Max := 100;
  eve_track.Frequency := 10;
END; {TULI.eve_randClick}

PROCEDURE TULI.eve_absClick(Sender: TObject);                   //im Dropdwownmenu wird absolut ausgewählt
BEGIN
  eve_track.min := 1;
  eve_track.Frequency := 1;
  eve_kind.Caption := 'absolut';
  eve_track.Max := clubsteranz;
END; {TULI.eve_absClick}

PROCEDURE TULI.eve_trackChange(Sender: TObject);                  //Track Bar der EVE-Anzahl wird verändert
BEGIN
  eve_label.Caption := IntToStr(eve_track.Position);
END; {TULI.eve_trackChange}

PROCEDURE TULI.delay_trackChange(Sender: TObject);                            //Delay-Track wird verändern
BEGIN
  delay_label.Caption := IntToStr(delay_track.Position);
END; {TULI.delay_trackChange}

PROCEDURE TULI.symp_randClick(Sender: TObject);                           //Sympathiewerte per Zufall füllen
VAR
  i: BYTE;
BEGIN
  FOR i := 0 TO clubsteranz - 1 DO clubsters[i].randomize(Sender);
END; {TULI.symp_randClick}

PROCEDURE TULI.symp_relClick(Sender: TObject);                           //Sympathiewerte realistisch füllen
VAR
  x, y, symp: BYTE;
BEGIN
  FOR x := 0 TO clubsteranz - 2 DO
    FOR y := x + 1 TO clubsteranz - 1 DO BEGIN
      symp := random(3) - 1;
      clubsters[x].sympathiewerte[y] := symp;                   //Zwei gegenüberliegende Clubster bokommen
      clubsters[y].sympathiewerte[x] := symp;                                 //die gleichen Sympathiewerte
    END;
END; {TULI.symp_relClick}

PROCEDURE TULI.clubsters_resetClick(Sender: TObject);              //Zurücksetzen der Verlässligkeitswerte
VAR
  i, k: BYTE;
BEGIN
  FOR i := 0 TO clubsteranz - 1 DO
    FOR k := 0 TO clubsteranz - 1 DO clubsters[i].verlass[k] := 0;
END; {TULI.clubsters_resetClick}

PROCEDURE TULI.verstrickungClick(Sender: TObject);        //Diagramm welches die Verhältnisse zwischen den
VAR                                                                                    //Clubsters verdeutlicht
  i, k: BYTE;
  w, w2, x, y: REAL;
BEGIN
  IF NOT (diagramm IS Tdiagramm) THEN application.CreateForm(Tdiagramm, diagramm);  //Öffnet das Diagrammf.
  diagramm.Visible := True;
  diagramm.Image1.Canvas.Rectangle(0, 0, 500, 500);
  FOR i := 0 TO clubsteranz - 1 DO BEGIN
    w := (i * 2 * PI) / clubsteranz;
    diagramm.Image1.Canvas.Pen.color;
    diagramm.Image1.Canvas.TextOut(round(240 + 220 * cos(w)), round(242 + 220 * sin(w)),
      IntToStr(i + 1));                                                    //Ausgabe der clubster Nummer
    IF (length(uli_auswertung) <> 0) AND (length(clubster_auswertung) <> 0) THEN
      diagramm.Image1.Canvas.TextOut(round(235 + 220 * cos(w)),
        round(258 + 220 * sin(w)), IntToStr(round(100 * clubster_auswertung[i].cyberin /
        length(uli_auswertung))) + '%');                 //Ausgabe der prozentualen Anzahl von cyberin-Tagen

    FOR k := i + 1 TO clubsteranz - 1 DO
      WITH diagramm.Image1.Canvas DO BEGIN
        MoveTo(round(250 + 200 * cos(w)), round(250 + 200 * sin(w)));
        w2 := (k * 2 * PI) / clubsteranz;
        x := 250 + 200 * cos(w2);
        y := 250 + 200 * sin(w2);

        CASE clubsters[k].sympathiewerte[i] OF -1: pen.color := clred;                    //wie findet k i
          0: pen.color := clyellow;
          1: pen.color := cllime;
        END;

        IF clubsters[i].verlass[k] < - 5 THEN pen.Style := psdot else     //wie kann sich i auf k verlassen
        IF clubsters[i].verlass[k] > 3 THEN pen.Width := 2;

        lineto(round((250 + 200 * cos(w) + x) / 2), round((250 + 200 * sin(w) + y) / 2));
                                                                  //Linie vom Clubster i zur Mitte(i,k)
        pen.style := pssolid;
        pen.Width := 1;

        IF clubsters[k].verlass[i] < - 5 THEN pen.Style := psdot else     //wie kann sich k auf i verlassen
        IF clubsters[k].verlass[i] > 3 THEN pen.Width := 2;

        CASE clubsters[i].sympathiewerte[k] OF -1: pen.color := clred;                    //wie findet i k
          0: pen.color := clyellow;
          1: pen.color := cllime;
```

```
        END;

        lineto(round(x), round(y));                                    //Linie von Mitte(i,k) zum Clubster k
        pen.style := pssolid;
        pen.width := 1;
      END;
  END;
END; {TULI.verstrickungClick}

PROCEDURE TULI.uebernehmenClick(Sender: TObject); //Dublizieren der Sende-Frage Stategien auf alle Clubster
VAR
  fragen, antworten: BOOLEAN;
  i, pers: BYTE;
BEGIN
  IF uebernehmen_art.Text = 'Fragen' THEN BEGIN
    fragen := True;
    antworten := False
  END ELSE IF uebernehmen_art.Text = 'Antworten' THEN BEGIN
    fragen := False;
    antworten := True
  END ELSE IF uebernehmen_art.Text = 'Fragen+Antworten' THEN BEGIN
    fragen := True;
    antworten := True
  END ELSE BEGIN
    fragen := False;
    antworten := False;
  END;

  FOR i := uebernehmen_zahl.Value TO clubsteranz - 1 DO BEGIN
    pers := i mod uebernehmen_zahl.Value;
    IF fragen THEN BEGIN
      clubsters[i].fragereihe.Items := clubsters[pers].fragereihe.Items;
      clubsters[i].max_fragen.Value := clubsters[pers].max_fragen.Value;
    END;
    IF antworten THEN BEGIN
      clubsters[i].zuv_symp.Value := clubsters[pers].zuv_symp.Value;
      clubsters[i].unz_symp.Value := clubsters[pers].unz_symp.Value;
      clubsters[i].zuv_neut.Value := clubsters[pers].zuv_neut.Value;
      clubsters[i].unz_neut.Value := clubsters[pers].unz_neut.Value;
      clubsters[i].zuv_unsy.Value := clubsters[pers].zuv_unsy.Value;
      clubsters[i].unz_unsy.Value := clubsters[pers].unz_unsy.Value;
      clubsters[i].sicherheit_t.Position := clubsters[pers].sicherheit_t.Position;
    END;
  END;
END; { TULI.uebernehmenClick}

PROCEDURE TULI.club_erstellenClick(Sender: TObject);          //Auswertungsdatei (nach Clubsters) erstellen
VAR
  f: textfile;
  i: WORD;
BEGIN
  club_erstellen.Enabled := False;
  assignfile(f, club_file.Text);
  rewrite(f);
  writeln(f, 'ClubsterNr.', chr(9), 'cyberin', chr(9), 'eingeweiht',         //Spaltenbezeichnung schreiben
    chr(9), 'richtig', chr(9), 'falsch', chr(9), 'Anfragen');
  FOR i := 0 TO clubsteranz - 1 DO WITH clubster_auswertung[i] DO        //Daten aus dem ARRAY in die Datei
      writeln(f, IntToStr(i + 1), chr(9), IntToStr(cyberin), chr(9),           //übertragen (Trennung: TAB)
        IntToStr(eingeweiht), chr(9), IntToStr(richtige), chr(9), IntToStr(falsche),
        chr(9), IntToStr(anfragen));
  closefile(f);

  setlength(clubster_auswertung, 0);                                    //Auswertungsarray zurücksetzen
END; {TULI.club_erstellenClick}

PROCEDURE TULI.uli_erstellenClick(Sender: TObject);               //Auswertungsdatei (nach Tagen) erstellen
VAR
  f: textfile;
  i: WORD;
BEGIN
  uli_erstellen.Enabled := False;
  assignfile(f, uli_file.Text);
  rewrite(f);
  writeln(f, 'TagNr.', chr(9), 'traffic', chr(9), 'cyberin', chr(9),         //Spaltenbezeichnung schreiben
    'eingeweiht', chr(9), 'richtig', chr(9), 'falsch', chr(9), 'Anfragen');

  FOR i := 0 TO length(uli_auswertung) - 1 DO WITH uli_auswertung[i] DO
      writeln(f, IntToStr(i + 1), chr(9), IntToStr(traffic), chr(9),     //Daten aus dem ARRAY in die Datei
        IntToStr(cyberin), chr(9), IntToStr(eingeweiht), chr(9), IntToStr(richtige),
        chr(9), IntToStr(falsche), chr(9), IntToStr(anfragen));               //übertragen (Trennung: TAB)
  closefile(f);

  setlength(uli_auswertung, 0);                                         //Auswertungsarray zurücksetzen
END; {TULI.uli_erstellenClick}

END.
```

Clubster1.pas

```
UNIT clubster1;

INTERFACE

USES
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, nachrichten, Menus, Grids, Spin, ExtCtrls, ComCtrls;
```

```
TYPE
  Tclubster = CLASS(TForm)
    MainMenu1: TMainMenu;
    antw: TMenuItem;
    frag: TMenuItem;
    vars: TMenuItem;
    symp: TMenuItem;
    verl: TMenuItem;
    anzeigen1: TMenuItem;
    random1: TMenuItem;
    StringGrid1: TStringGrid;
    Memo1: TMemo;
    antworten: TPanel;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    StaticText4: TStaticText;
    zuv_symp: TSpinEdit;
    unz_symp: TSpinEdit;
    zuv_neut: TSpinEdit;
    unz_neut: TSpinEdit;
    zuv_unsy: TSpinEdit;
    unz_unsy: TSpinEdit;
    Button1: TButton;
    StaticText5: TStaticText;
    sicherheit_t: TTrackBar;
    PopupMenu1: TPopupMenu;
    korrektgarnicht1: TMenuItem;
    korrektfalschoderrichtig1: TMenuItem;
    Beispiel31: TMenuItem;
    fragen: TPanel;
    fragereihe: TListBox;
    Button2: TButton;
    fragearten: TComboBox;
    StaticText6: TStaticText;
    fragearten_anz: TSpinEdit;
    Bevel1: TBevel;
    Bevel2: TBevel;
    StaticText8: TStaticText;
    max_fragen: TSpinEdit;
    alle_loeschen: TButton;
    aktiv_loeschen: TButton;
    Button3: TButton;
    PopupMenu2: TPopupMenu;
    anSympathische1: TMenuItem;
    anVerlsslicheunddannalle1: TMenuItem;

    PROCEDURE neuer_tag();                                          //werden von ULI ausgelöst
    PROCEDURE bearbeiten();

    FUNCTION getinput: nachricht;                                   //interne Prozeduren
    PROCEDURE send(inp: nachricht);
    PROCEDURE neuzeichnen();
    PROCEDURE wohin();

    PROCEDURE sympanz(Sender: TObject);
    PROCEDURE FormDeactivate(Sender: TObject);
    PROCEDURE randomize(Sender: TObject);
    PROCEDURE verlanz(Sender: TObject);
    PROCEDURE StringGrid1Click(Sender: TObject);
    PROCEDURE antwClick(Sender: TObject);
    PROCEDURE sicherheit_tChange(Sender: TObject);
    PROCEDURE antw_presets(Sender: TObject);
    PROCEDURE korrektgarnicht1Click(Sender: TObject);
    PROCEDURE korrektfalschoderrichtig1Click(Sender: TObject);
    PROCEDURE Beispiel31Click(Sender: TObject);
    PROCEDURE frag_hinzu(Sender: TObject);
    PROCEDURE alle_loeschenClick(Sender: TObject);
    PROCEDURE aktiv_loeschenClick(Sender: TObject);
    PROCEDURE frag_presets(Sender: TObject);
    PROCEDURE anSympathische1Click(Sender: TObject);
    PROCEDURE anVerlsslicheunddannalle1Click(Sender: TObject);
    PROCEDURE fragClick(Sender: TObject);

  PRIVATE
    uli_eve: boolean;                                               //true = von ULI benachrichtigt
    fragestunde: boolean;     //true = hat die Fragen schon in den Sendbuffer geschrieben wenn kein ULI_EVE
    beste_wahl: Tinhalt;                    //Denn Treffpunkt den der Clubster als wahrscheinlichsten annimmt
    sicherheit: real;                                  //Sicherheit die er für diesen Treffpunkt annimmt
    message_book: ARRAY OF Tinhalt;                     //
  PUBLIC
    auswertung: Tauswertung;               //Variable mit den Countern die für die auswertung wichtig sind
    meinenummer: byte;                                                  //die Nummer des Clubsters
    mitgliederanz: byte;                                          //die Gesamtamzahl der Clubster
    sendbuffer: ARRAY OF nachricht;                          //ARRAY mit den überlegten Fragen + Antworten
    getbuffer: ARRAY OF nachricht;                            //ARRAY mit den eingehenden SMS-Nachrichten
    verlass: ARRAY OF integer;                  //ARRAY der Verlässligkeistwerte (Grösser = verlässlicher)
    sympathiewerte: ARRAY OF shortint;                //ARRAY der Sympathiewerte (-1:uns., 0:neut, 1:symp)
  END;

IMPLEMENTATION
{$R *.DFM}

PROCEDURE Tclubster.send(inp: nachricht);    //velängerst den sendbuffer und fügt eine neue Nachricht hinzu
VAR l: byte;
BEGIN
  l := length(sendbuffer);
  setlength(sendbuffer, l + 1);
  sendbuffer[l] := inp;
END; {Tclubster.send}

FUNCTION Tclubster.getinput: nachricht;     //Nimmt das erste Element vom getbuffer und verschiebt den Rest
```

```
VAR i: byte;
BEGIN
  getinput := getbuffer[0];
  FOR i := 1 TO length(getbuffer) DO getbuffer[i - 1] := getbuffer[i];
  setlength(getbuffer, length(getbuffer) - 1);
END; {Tclubster.getinput}

PROCEDURE Tclubster.wohin();                              //Symuliert den Clubster bei der Wahl des Treffpunkts
VAR i, k, messageanz: byte;
  warscheinlichkeit: real;
  antwort_moeglich: ARRAY OF Tinhalt;                    //alle Treffpunkte die in den Antworten enthalten sind
  antwort_moeglich_anz: ARRAY OF byte;                              //wie oft wurde der Treffpunkt übermittelt
  vorhanden: boolean;
BEGIN
  messageanz := 0;
  FOR i := 0 TO mitgliederanz - 1 DO
    IF message_book[i].art = [eve] THEN BEGIN                                    //Durchsucht alle Antworten
      inc(messageanz);
      vorhanden := false;
      IF length(antwort_moeglich) <> 0 THEN
        FOR k := 0 TO length(antwort_moeglich) - 1 DO                  //ist der Treffpunkt schon vorhanden
          IF (message_book[i].zeit = antwort_moeglich[k].zeit) AND
            (message_book[i].ort = antwort_moeglich[k].ort) THEN BEGIN
            vorhanden := true;
            inc(antwort_moeglich_anz[k]);
          END;
      IF NOT (vorhanden) THEN BEGIN                                  //macht einen neuen Treffpunkt im ARRAY
        setlength(antwort_moeglich, length(antwort_moeglich) + 1);
        setlength(antwort_moeglich_anz, length(antwort_moeglich));
        antwort_moeglich[length(antwort_moeglich) - 1] := message_book[i];
        antwort_moeglich_anz[length(antwort_moeglich) - 1] := 1;
      END;
    END;
  IF messageanz < 3 THEN messageanz := 3;                  //erst bei 3 gleichen Nachrichten 100% Sicherheit
  k := 0;
  warscheinlichkeit := antwort_moeglich_anz[k] / messageanz;

  FOR i := 1 TO length(antwort_moeglich_anz) - 1 DO           //Wählt den wahrscheinlichsten Treffpunkt aus
    IF warscheinlichkeit > antwort_moeglich_anz[i] / messageanz THEN BEGIN
      k := i;
      warscheinlichkeit := antwort_moeglich_anz[k] / messageanz;
    END;

  sicherheit := warscheinlichkeit;
  beste_wahl := antwort_moeglich[k];

  memo1.Color := rgb(255, 100 + round(150 * sicherheit), 0);     //gibt dem Clubster eine Sicherheits-Farbe
END; {Tclubster.wohin}

PROCEDURE Tclubster.neuer_tag();
VAR i: byte;                                              //Stetzt die Werte für einen neuen Tag zurück
BEGIN
  uli_eve := false;
  fragestunde := false;
  sicherheit := 0;
  beste_wahl.ort := 0;
  beste_wahl.zeit := 0;
  memo1.Color := clBtnFace;
  setlength(sendbuffer, 0);
  setlength(getbuffer, 0);
  setlength(sympathiewerte, mitgliederanz);
  setlength(verlass, mitgliederanz);
  setlength(message_book, mitgliederanz);
  FOR i := 0 TO mitgliederanz - 1 DO message_book[i].art := [];
  IF visible THEN neuzeichnen;
END; {Tclubster.neuer_tag}

PROCEDURE Tclubster.bearbeiten();      //lässt den Clubster eine Nachricht lesen und eine/mehrere Versenden
VAR eingang, richtig, falsch, anfrage: nachricht;
  i, frage_anz, person, anzahl_gesamt, anzahl_typ, tries: byte;
  antwwahrsch: integer;
  frage_typ, frage_anzahl: STRING;
  nachrichtanperson: boolean;
BEGIN
  eingang.inhalt.art := [];
  IF length(getbuffer) <> 0 THEN eingang := getinput;       //Holt sich einen eingehende SMS aus dem Buffer

  IF eingang.inhalt.art = [eve] THEN BEGIN                                  //Behadlung der EVE-Nachrichten
    IF eingang.von = 255 THEN BEGIN                                       //Wenn die Nachricht von ULI kommt
      uli_eve := true;
      sicherheit := 1;
      beste_wahl := eingang.inhalt;
      memo1.Color := cllime;
    END ELSE BEGIN                                          //Wenn die Nachricht von einem anderen Clubster kommt
      message_book[eingang.von] := eingang.inhalt;
      wohin();
    END;
  END;


  IF eingang.inhalt.art = [peve] THEN BEGIN                                 //Behandlung der PEVE-Nachricht
    IF (beste_wahl.ort <> eingang.inhalt.ort) OR (beste_wahl.zeit <> eingang.inhalt.zeit)
      THEN BEGIN                                                            //War die Entscheidung korrekt?
      memo1.Color := clred; auswertung.wahl := false;
    END
    ELSE auswertung.wahl := true;
    auswertung.eingeweiht := uli_eve;
    auswertung.falsche := 0;
    auswertung.richtige := 0;
    auswertung.anfragen := 0;
    FOR i := 0 TO mitgliederanz - 1 DO BEGIN                          //Durchsucht das Messagebook und
      IF message_book[i].art = [eve] THEN BEGIN
```

```
        inc(auswertung.anfragen);
        IF (message_book[i].ort = eingang.inhalt.ort) AND                          //richtig SMS: Vertrauen+2
          (message_book[i].zeit = eingang.inhalt.zeit) THEN BEGIN
          inc(verlass[i], 2);
          inc(auswertung.richtige);
        END
        ELSE BEGIN                                                                 //falsche SMS: Vertrauen-3
          dec(verlass[i], 3); inc(auswertung.falsche);
        END
      END ELSE IF message_book[i].art = [wuw] THEN BEGIN                          //keine Antwort: Verlass -1
        inc(auswertung.anfragen);
        //dec(verlass[i]);
      END
    END;
    setlength(sendbuffer, 0);                                                      //Löscht die beiden Buffer
    setlength(getbuffer, 0);
  END;

          //Wenn der Clubster nicht antworten kann verschiebt er die SMS vom Anfang zum Ende des getbuffers
  IF (eingang.inhalt.art = [wuw]) AND NOT (uli_eve)
    AND (sicherheit * 100 < sicherheit_t.Position) AND (sicherheit_t.Position < 101) THEN BEGIN
    setlength(getbuffer, length(getbuffer) + 1);
    getbuffer[length(getbuffer) - 1] := eingang;
  END;

                                                                                   //Clubster anwortet auf WUW-Anfragen
  IF (eingang.inhalt.art = [wuw]) AND (uli_eve OR (sicherheit * 100 >= sicherheit_t.Position)) THEN BEGIN
    richtig.von := eingang.von;                                  //Bildung einer Richtigen und falschen Antwort
    falsch.von := eingang.von;
    richtig.inhalt := beste_wahl;
    falsch.inhalt := beste_wahl;
    REPEAT
      falsch.inhalt.ort := random(5) + 1;
      falsch.inhalt.zeit := random(5) + 1;
    UNTIL (falsch.inhalt.zeit <> beste_wahl.zeit) OR (falsch.inhalt.zeit <> beste_wahl.ort);
    IF verlass[eingang.von] >= 0 THEN                                //Entscheiden welcher Antworten-Typ für den
      CASE sympathiewerte[eingang.von] OF                                          //fragenden Clubster zutrifft
        -1: antwwahrsch := zuv_unsy.value;
        0: antwwahrsch := zuv_neut.value;
        1: antwwahrsch := zuv_symp.value;
      END ELSE
      CASE sympathiewerte[eingang.von] OF
        -1: antwwahrsch := unz_unsy.value;
        0: antwwahrsch := unz_neut.value;
        1: antwwahrsch := unz_symp.value;
      END;                                       //Verschickt eine Nachricht wenn antwwahrsch nicht -10 ist
    IF antwwahrsch >= 0 THEN IF random(100) > antwwahrsch THEN send(falsch) ELSE send(richtig);
  END;

        //Wenn der Clubster zu beginn kein ULI_eve bekommt, ueberlegt er sich alle Fragen und stellt sie an
  IF NOT (uli_eve) AND NOT (fragestunde) THEN BEGIN                                //den Anfang des Sendbuffers
    anfrage.inhalt.art := [wuw];
    fragestunde := true;
    anzahl_gesamt := 0;
    FOR i := 0 TO fragereihe.Items.count - 1 DO BEGIN                  //Durchforstet die alle Listeneinträge
      frage_typ := copy(fragereihe.Items.Strings[i], 0, Pos(',', fragereihe.Items.Strings[i]) - 1);
      frage_anzahl := copy(fragereihe.Items.Strings[i], Pos(',', fragereihe.Items.Strings[i]) + 1, 500);
      IF frage_anzahl = 'alle' THEN frage_anz := mitgliederanz ELSE frage_anz := strtoint(frage_anzahl);
      anzahl_typ := 0;
      REPEAT                              //Schleife für die Anzahl der Clubser die unter diese Kategorie fallen
        tries := 0;
        REPEAT
          nachrichtanperson := false;
          inc(tries);
          REPEAT
            person := random(mitgliederanz);                          //Sucht sich per Zufall eine Person aus
          UNTIL (person <> meinenummer - 1) AND (message_book[person].art <> [wuw]);
          IF frage_typ = 'alle' THEN nachrichtanperson := true; //Testet ob die Person die Vorgaben erfüllt
          IF (frage_typ = 'zuv.') AND (verlass[person] > 0) THEN nachrichtanperson := true;
          IF (frage_typ = 'symp.') AND (sympathiewerte[person] = 1) THEN nachrichtanperson := true;
          IF (frage_typ = 'neutral') AND (sympathiewerte[person] = 0) THEN nachrichtanperson := true;
          IF (frage_typ = 'symp. + neutral') AND (sympathiewerte[person] <> -1) THEN
            nachrichtanperson := true;
          IF (frage_typ = 'symp. + zuv.') AND (sympathiewerte[person] = 1) AND (verlass[person] > 0)
            THEN nachrichtanperson := true;
          IF (frage_typ = 'neutral + zuv.') AND (sympathiewerte[person] = 0) AND (verlass[person] > 0)
            THEN nachrichtanperson := true;
          IF (frage_typ = 'symp. oder neutral + zuv.') AND (sympathiewerte[person] <> -1) AND
            (verlass[person] > 0) THEN nachrichtanperson := true;
        UNTIL nachrichtanperson OR (tries > 100);
        IF (tries <> 101) AND nachrichtanperson AND (max_fragen.Value > anzahl_gesamt) THEN BEGIN
          anfrage.von := person;                                      //Schickt eine Angfrage an den Clubster
          message_book[person] := anfrage.inhalt;
          send(anfrage);
          inc(anzahl_gesamt);
          inc(anzahl_typ);
        END;
    UNTIL (max_fragen.Value <= anzahl_gesamt) OR (anzahl_typ >= frage_anz) OR (tries > 100);
   END;
  END;

  IF visible THEN neuzeichnen;
END; {Tclubster.bearbeiten}

PROCEDURE Tclubster.neuzeichnen();    //Erneuert en Text den die Clubster im deaktiviertem Zustand anzeigen
BEGIN
   memo1.lines.Strings[0] :=
    'Sendbuff.: ' + inttostr(length(sendbuffer)) + ' | Getbuf.: ' + inttostr(length(getbuffer));
  memo1.lines.Strings[1] :=
    'Sicherheit: ' + inttostr(round(sicherheit * 100)) + '%';
  memo1.lines.Strings[2] :=
    'Ort Nr.: ' + inttostr(beste_wahl.ort) + ' | Zeit Nr.:' + inttostr(beste_wahl.zeit);
```

```
END; {Tclubster.neuzeichnen}

PROCEDURE Tclubster.sympanz(Sender: TObject);    //Zeigt die Sympathiewerte in der Tabelle der Clubsters an
VAR i: byte;
BEGIN
  memo1.visible := false;
  antworten.Visible := false;
  fragen.Visible := false;
  stringgrid1.Visible := true;
  height := 183;
  stringgrid1.Cells[0, 0] := 'clubster Nr.';
  stringgrid1.Cells[1, 0] := 'sympatisch?';
  stringgrid1.rowcount := mitgliederanz + 1;
  FOR i := 1 TO mitgliederanz DO BEGIN
    stringgrid1.Cells[0, i] := inttostr(i);
    IF i = meinenummer THEN stringgrid1.Cells[1, i] := '-' ELSE
      CASE sympathiewerte[i - 1] OF
        0: stringgrid1.Cells[1, i] := 'neutral';
        1: stringgrid1.Cells[1, i] := 'symp.';
        -1: stringgrid1.cells[1, i] := 'unsymp';
      END;

  END;
END; {Tclubster.sympanz}

PROCEDURE Tclubster.FormDeactivate(Sender: TObject);      //Zeigt den Satustext an und Scheibt den Clubster
BEGIN                                                                    //wieder an seine Position
  height := 74;
  memo1.Visible := true;
  stringgrid1.Visible := false;
  antworten.Visible := false;
  fragen.Visible := false;
  top := uli_height + ((meinenummer - 1) DIV (screen.Width DIV width)) * height;
  left := ((meinenummer - 1) MOD (screen.Width DIV width)) * width;
END; {Tclubster.FormDeactivate}

PROCEDURE Tclubster.randomize(Sender: TObject);                          //Vergiebt zufällige Sympathiewerte
VAR i: byte;
BEGIN
  FOR i := 0 TO mitgliederanz - 1 DO sympathiewerte[i] := random(3) - 1;
  stringgrid1.Cells[0, 0] := 'clubster Nr.';
  stringgrid1.Cells[1, 0] := 'sympatisch?';
  stringgrid1.rowcount := mitgliederanz + 1;
  FOR i := 1 TO mitgliederanz DO BEGIN
    stringgrid1.Cells[0, i] := inttostr(i);
    IF i = meinenummer THEN stringgrid1.Cells[1, i] := '-' ELSE
      CASE sympathiewerte[i - 1] OF
        0: stringgrid1.Cells[1, i] := 'neutral';
        1: stringgrid1.Cells[1, i] := 'symp.';
        -1: stringgrid1.cells[1, i] := 'unsymp';
      END
  END;
END; {Tclubster.randomize}

PROCEDURE Tclubster.verlanz(Sender: TObject);         //Zeigt eine Tabelle mit den Verlässligkeitswerten an
VAR i: byte;
BEGIN
  fragen.Visible := false;
  memo1.visible := false;
  antworten.Visible := false;
  stringgrid1.Visible := true;
  height := 183;
  stringgrid1.Cells[0, 0] := 'clubster Nr.';
  stringgrid1.Cells[1, 0] := 'Verlässligkeit';
  stringgrid1.rowcount := mitgliederanz + 1;
  FOR i := 1 TO mitgliederanz DO BEGIN
    stringgrid1.Cells[0, i] := inttostr(i);
    IF i = meinenummer THEN stringgrid1.Cells[1, i] := '-' ELSE
      stringgrid1.Cells[1, i] := inttostr(verlass[i - 1]);
  END;
END; {Tclubster.verlanz}

PROCEDURE Tclubster.StringGrid1Click(Sender: TObject);  //Verändert den Sympathiewert wenn man ihn anklickt
VAR i: byte;
BEGIN
  i := stringgrid1.row - 1;
  CASE sympathiewerte[i] OF
    -1: sympathiewerte[i] := 0;
    0: sympathiewerte[i] := 1;
    1: sympathiewerte[i] := -1;
  END;
  sympanz(Sender);
END; {Tclubster.StringGrid1Click}

PROCEDURE Tclubster.antwClick(Sender: TObject);//Zeigt das Pannel an, welches die Antwortstrategie festlegt
BEGIN
  antworten.Visible := true;
  height := 183;
  stringgrid1.Visible := false;
  memo1.Visible := false;
  fragen.Visible := false;
END; {Tclubster.antwClick}

PROCEDURE Tclubster.sicherheit_tChange(Sender: TObject);                //Schieber auf dem Antworten-Pannel
BEGIN                                                                                  //Schreibt anstatt 101% ULI
  IF sicherheit_t.position = 101 THEN statictext5.Caption := 'Sicherheit: ULI' ELSE
    statictext5.Caption := 'Sicherheit: ' + inttostr(sicherheit_t.position) + '%';
END; {Tclubster.sicherheit_tChange}

PROCEDURE Tclubster.antw_presets(Sender: TObject);     //Öffnet das Antwortenpresetauswahldropdownmenue :-D
BEGIN
  popupmenu1.Popup(mouse.CursorPos.x, mouse.CursorPos.y);
```

```
END; {Tclubster.antw_presets}

PROCEDURE Tclubster.korrektgarnicht1Click(Sender: TObject);              //Antwortenpreset: Vorgabe Nummer1
BEGIN
  zuv_symp.Value := 100;
  unz_symp.Value := 100;
  zuv_neut.Value := -10;
  unz_neut.value := -10;
  zuv_unsy.Value := -10;
  unz_unsy.Value := -10;
  sicherheit_t.Position := 101;
END; {Tclubster.korrektgarnicht1Click}

PROCEDURE Tclubster.korrektfalschoderrichtig1Click(Sender: TObject);     //Antwortenpreset: Vorgabe Nummer2
BEGIN
  zuv_symp.Value := 100;
  unz_symp.Value := 100;
  zuv_neut.Value := 50;
  unz_neut.value := 50;
  zuv_unsy.Value := 0;
  unz_unsy.Value := 0;
  sicherheit_t.Position := 101;
END; {Tclubster.korrektfalschoderrichtig1Click}

PROCEDURE Tclubster.Beispiel31Click(Sender: TObject);                    //Antwortenpreset: eigenes Beispiel
BEGIN
  zuv_symp.Value := 100;
  unz_symp.Value := 90;
  zuv_neut.Value := 90;
  unz_neut.value := -10;
  zuv_unsy.Value := 0;
  unz_unsy.Value := 0;
  sicherheit_t.Position := 60;
END; {Tclubster.Beispiel31Click}

PROCEDURE Tclubster.frag_hinzu(Sender: TObject);                         //Hängt einen Fragetyp an die Liste
BEGIN
  IF fragearten_anz.Value = 0 THEN fragereihe.Items.Add(fragearten.Text + ',alle')
  ELSE fragereihe.Items.Add(fragearten.Text + ',' + inttostr(fragearten_anz.Value));
END; {Tclubster.frag_hinzu}

PROCEDURE Tclubster.alle_loeschenClick(Sender: TObject);             //Löscht alle Fragetypen aus der Liste
BEGIN
  fragereihe.Items.Clear;
END; {Tclubster.alle_loeschenClick}

PROCEDURE Tclubster.aktiv_loeschenClick(Sender: TObject);             //Löscht das angewählte Listenelement
VAR i: byte;
BEGIN
  FOR i := 0 TO fragereihe.Items.Count - 1 DO BEGIN
    IF (i < fragereihe.Items.Count) AND fragereihe.Selected[i] THEN BEGIN
      fragereihe.Items.Delete(i);
    END;
  END;
END; {Tclubster.aktiv_loeschenClick}

PROCEDURE Tclubster.frag_presets(Sender: TObject);  //Öffnet ein Dropdownmenü für die Fragestrategiepresets
BEGIN
  popupmenu2.Popup(mouse.CursorPos.x, mouse.CursorPos.y);
END;  {Tclubster.frag_presets}

PROCEDURE Tclubster.anSympathische1Click(Sender: TObject);                   //Fragestrategie: Vorgabe Nr.1
BEGIN
  fragereihe.Items.Clear;
  fragereihe.Items.Add('symp.,alle');
END; {Tclubster.anSympathische1Click}

PROCEDURE Tclubster.anVerlsslicheunddannalle1Click(Sender: TObject);         //Fragestrategie: Vorgabe Nr.2
BEGIN
  fragereihe.Items.Clear;
  fragereihe.Items.Add('zuv.,alle');
  fragereihe.Items.Add('alle,5');
END; {Tclubster.anVerlsslicheunddannalle1Click}

PROCEDURE Tclubster.fragClick(Sender: TObject);    //Öffnet das Pannel, welches die Fragestrategie festlegt
BEGIN
  fragen.Visible := true;
  height := 233;
  stringgrid1.Visible := false;
  memo1.Visible := false;
  antworten.Visible := false;
END; {Tclubster.fragClick}

END.
```

Diagramm1.pas

```
UNIT diagramm1;

INTERFACE

USES
  Windows, SysUtils, Graphics, Controls, Forms, ExtCtrls, Classes;

TYPE
  Tdiagramm = CLASS(TForm)
    Image1: TImage;
  PRIVATE
    { Private-Deklarationen }
  PUBLIC
    { Public-Deklarationen }
  END;

VAR
  diagramm: Tdiagramm;

IMPLEMENTATION

{$R *.DFM}

END.
```