EUMEL

Extendable multi User Microprocessor ELAN-system

Anwendersoftwareklasse 2

Hamster – Modell

Hochschul-
Rechen-
Zentrum

Universität Bielefeld

GMD

1. Einsatzbereich und Grenzen des Hamstermodells 1

1.1 Kurzbeschreibung des Hamstermodells 1

2. Benutzung des EUMEL – Hamsterpakets 2

2.1 Basis – Sprachelemente der Hamstersprache 3
2.1.1 Basis – Befehle der Hamstersprache 3
2.1.2 Basis – Tests der Hamstersprache 3
2.1.3 Zusätzliche Sprachelemente 4

2.2 Erstellen und Korrigieren von Hamsterprogrammen 4

2.3 Erstellen von Hamsterlandschaften 5
2.3.1 Landschaftsnamen, Form der Landschaft 6
2.3.2 Tastenfunktionen des Landschaftseditors 7
2.3.3 Startposition des Hamsters, Backentaschen 8

2.4 Ausführen von Hamsterprogrammen 8
2.4.1 Wirkung der Basis – Befehle und – Tests 9
2.4.2 Beschleunigen und Verzögern des Hamsterlaufs 11
2.4.3 Programmabbruch durch Tastendruck 11
2.4.4 Tasteneingabe und – auswertung 11
2.4.5 Angabe von Landschaftsnamen im Hamsterprogramm 12

3. Installation des Hamstermodells 12

1. Einsatzbereich und Grenzen des Hamstermodells

Das Hamstermodell soll dazu dienen, die Grundelemente der Programmierung einfach und spielend zu erlernen, ohne die vielen 'Widerwärtigkeiten der harten DV – Praxis' am Anfang berücksichtigen zu müssen. Daher steht am Anfang nicht der Computer im Mittelpunkt, sondern ein Modell, welches so einfach und klein ist, daß ein Anfänger schon nach einer halben Stunde in der Lage ist, das gesamte Modell so weit zu überblicken, daß er sich selbst einfache Aufgaben stellen und diese lösen kann.

Das Modell ist so offen und flexibel, daß es sich mit wachsendem Wissen und Können erweitern läßt. Praktisch jedes Element der Programmiersprache ELAN läßt sich hinzufügen, so daß fortgesetzte Erweiterungen schließlich in der Sprache ELAN im vollen Umfang münden.

Das Hamstermodell ist angelegt für Programmieren und Ausführen mit Bleistift auf kariertem Papier.

Auf jeden Fall soll so begonnen werden. Die Mechanismen 'Computer', 'Bild – schirm' und 'Tastatur' lenken am Anfang vom den Grundelementen der Pro – grammierung ab! Erst wenn die Basis – Befehle, Basis – Tests, das Bilden eigener benannter Anweisungen durch Refinements und die einfachen Kontrollanweisun – gen Fallunterscheidung (IF – THEN – ELSE – END IF) und Wiederholung (WHILE – REPEAT – END REPEAT) hinreichend vertraut sind, ist es sinnvoll, umfangreichere Hamster – Programmieraufgaben anzufassen. Diese werden dann mit Hilfe des EUMEL – Hamsters geprüft.

1.1 Kurzbeschreibung des Hamstermodells

Das einzige Wesen, welches existiert, ist ein Hamster mit der Fähigkeit, vier verschiedene Befehle auszuführen:

```
- vor         (gehe einen Schritt nach vorn)
- links um    (drehe dich um 90 Grad nach links)
- nimm        (nimm da, wo Du stehst, ein Korn auf)
- gib         (lege da, wo Du stehst, ein Korn ab)
```

Das Territorium, auf dem der Hamster lebt, ist eine gekachelte Ebene. Auf den Kacheln können Weizenkörner liegen. Auf einer Kachel steht der Hamster. Damit die Welt interessanter wird, können einige Kacheln blockiert sein. Der Hamster kann solche Kacheln nicht betreten.

Die Befehle 'nimm', 'gib' und 'vor' können nicht uneingeschränkt ausgeführt werden. Daher muß der Hamster die Fähigkeit haben, eine Kachel weit zu 'fühlen'. Sonst würden 'Laufzeitfehler' passieren.

Wir definieren für den Hamster drei Testbefehle:

- `vorn frei`
- `korn da`
- `backen leer`

Ein Testbefehl liefert immer einen Wahrheitswert und dient zum Einsatz in Fall – unterscheidungen und Wiederholungen.

Hamsterprogramme können geschrieben werden, indem Basisbefehle, durch Refinements definierte Anweisungen und Kontrollanweisungen der ELAN – Syntax entsprechend aneinandergereiht werden.

2. Benutzung des EUMEL – Hamsterpakets

Das Ausführen umfangreicher oder komplizierter Hamsterprogramme allein auf karriertem Papier wird schnell langweilig und unsicher.

Das im EUMEL realisierte Hamstermodell hat die Aufgabe, an diesem Punkt die Prüfarbeit zu übernehmen. Geprüft wird dabei zunächst die syntaktische Kor – rektheit. Bei der Ausführung der Hamsteroperationen als Bewegungen auf dem Bildschirm kann der Hamsterprogrammierer dann selbst überprüfen, ob das Programm das gewünschte leistet. (Es wäre nicht im Geiste des Hamstermodells, wollte man mit dem Softwaremodell beginnen, ohne vorher ausgiebig mit Bleistift und Papier Hamsterprobleme bearbeitet zu haben!)

2.1 Basis – Sprachelemente der Hamstersprache

2.1.1 Basis – Befehle der Hamstersprache

Es gibt in der Hamstersprache vier Basisbefehle:

```
PROC vor       (* gehe eine Kachel in Blickrichtung nach vorn *)
PROC links um (* drehe Dich eine Vierteldrehung nach links   *)
PROC nimm      (* hebe da, wo Du stehest, ein Korn auf        *)
PROC gib       (* lege da, wo Du stehst, ein Korn ab          *)
```

Aus diesen werden mit Hilfe der in ELAN üblichen Kontrollkonstrukten (Folge, Wiederholung und Fallunterscheidung) Hamsterprogramme konstruiert. Von Anfang an soll von der Möglichkeit, neue Anweisungen durch Refinements zu bilden, ausgiebig Gebrauch gemacht werden. Darüberhinaus darf jedes ELAN – Sprachelement (Prozedur, Paket, Variable, Konstante, Zuweisung, Kommentar, ROW, STRUCT, TYPE, usw.) hinzugezogen werden. Insbesondere darf auch jede Standardprozedur verwendet werden. Man beachte aber, daß Prozeduren, welche Ausgabe auf den Bildschirm leisten, die Hamsterlandschaft zerstören.

2.1.2 Basis – Tests der Hamstersprache

Zur Vermeidung von Programmabbrüchen dienen Tests. Es gibt drei solche Basis – Tests:

```
BOOL PROC vorn frei
BOOL PROC korn da
BOOL PROC backen leer
```

Die Tests werden in Fallunterscheidungen (IF – THEN – ELSE – END IF) und Wiederholungen (WHILE – REPEAT – END REPEAT) benutzt.

Man kann mittels Refinements aus Basisbefehlen und Basis – Tests leicht weitere Tests bilden. Hierbei muß aber beachtet werden, daß meistens Seiteneffekte auftreten, welche beseitigt werden sollten.

Beispiel: aus 'links um' und 'vorn frei' kann man kann man leicht 'links frei' bilden.

Damit der Hamster nach diesem Test aber wieder genau so steht wie vorher, wird die Konstruktion ein wenig umständlich und unschön:

```
links frei:
    links um;
    IF vorn frei
      THEN rechts um; TRUE
      ELSE rechts um; FALSE
    FI.
rechts um:
    links um; links um; links um.
```

2.1.3 Zusätzliche Sprachelemente

Über die Basis – Befehle hinaus gibt es zwei weitere Prozeduren, welche vor allem für fortgeschrittene Anwendungen gute Dienste leisten können:

TEXT PROC taste
: liefert jeweils ein eingetipptes Zeichen. Jedes Zeichen bis auf ESC kann geliefert werden (siehe 2.4.4)

PROC landschaft (TEXT CONST landschaftsname)
: ermöglicht es, das Erfragen der Landschaft beim Programmlauf zu unterdrücken. Der Hamster kann also, falls die angegebene Landschaft existiert, sofort loslaufen (siehe 2.4.5).

2.2 Erstellen und Korrigieren von Hamsterprogrammen

Hamsterprogramme werden mit dem EUMEL – Editor geschrieben und korrigiert. Von Anfang an soll auf eine übersichtliche Form geachtet werden. Dazu beachte man die Faustregeln:

- verwende viel Arbeit auf die Auswahl treffender Verbalisierungen

- ein Programm ist zwar eine Folge von Zeichen. Aber diese sollten strukturiert sein! Nutze die Zweidimensionalität des Bildschirms

- mache die Zeilen nicht zu lang, verwende Leerzeilen und Einrückungen

- erläutere schwierige Entwurfsentscheidungen durch Kommentare

2.3 Erstellen von Hamsterlandschaften

Hamsterlandschaften werden mit einem speziellen Landschaftseditor erstellt. Natürlich sollen die einzelnen Felder möglichst 'quadratisch' dargestellt werden. Da die Bildschirmschreibstellen meist viel höher als breit sind, werden für eine 'Kachel' zwei Schreibstellen verwendet. Benutzung des EUMEL – Editors zur Erstellung von Hamsterlandschaften würde komplizierte Prüfungen der Landschaftsgestaltung mit sich bringen. Daher wird ein (einfacher) spezieller Landschaftseditor bereitgestellt, mit dem man nur 'korrekte' Landschaften erstellen kann.

Der Landschaftseditor wird bei Ausführen des ersten Basis – Befehls gestartet. Er erfragt den Namen der zu behandelnden Landschaft. Man kann einen beliebigen Namen angeben. Wenn eine Landschaft mit diesem Namen bereits existiert wird sie zum Verändern angeboten. Andernfalls wird eine neue, leere Landschaft angeboten.

Man kann den Landschaftseditor aber auch unabhängig von der Ausführung eines Hamsterprogramms benutzen, z.B. vor Erstellen eines Hamsterprogramms. Dann gibt man in der Task das Kommando 'landschaft(name)'. Natürlich muß hier ein Landschaftsname als Parameter angegeben werden.

2.3.1 Landschaftsnamen, Form der Landschaft

Landschaftsnamen können aus Buchstaben, Ziffern und Sonderzeichen bestehen. Hamsterlandschaften können nicht mit dem EUMEL – Editor bearbeitet werden. Mit dem EUMEL – Editor lassen sich nämlich auch formal nicht korrekte Land – schaften erzeugen. Der Hamsterprozessor ist aber darauf angewiesen, daß die verwendete Landschaft immer korrekt ist.

Sperrung gegen den EUMEL – Editor wird dadurch erreicht, daß die Land – schaftsdatei den Datenraumtyp 1001 bekommt. Dieser wird vom EUMEL – Editor nicht akzeptiert. Außerdem wird vor den Landschaftsnamen intern das Präfix "Land:" gehängt.

Darstellung einer Ebene auf dem Bildschirm ist wegen deren Unbegrenztheit nicht einfach. Daher sind die Hamsterlandschaften Ausschnitte aus einer gekachelten Ebene mit 24 mal 40 Kacheln.

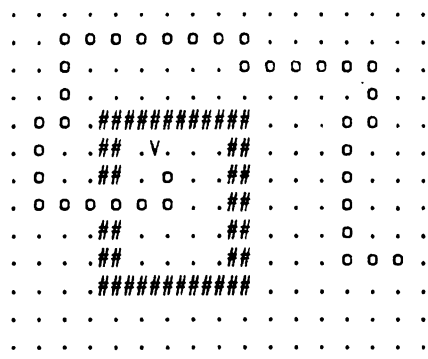
Eine Kachel kann sein:

Leere Kachel	Blank + Punkt	(" .")
Kornkachel	Blank + kleines o	(" o")
Hindernis	zwei Nummernzeichen	("##")

in dieser Landschaft steht auf einer Kachel der Hamster:

"A"	Richtung oben
" > "	Richtung rechts
"V"	Richtung unten
" < "	Richtung links

Beispiel: Ausschnitt aus einer Landschaft:



2.3.2 Tastenfunktionen des Landschaftseditors

Während der Landschaftsgestaltung wirken folgende Tasten:

h . . .	halt, beende die Lanschaftsgestaltung
# . . .	setze ein Hindernis und gehe ein Feld weiter
LEERTASTE	leere das Feld und gehe ein Feld weiter
g . . .	lege ein Korn ab
n . . .	nimm ein Korn auf (falls hier Körner liegen)
z . . .	zeige, wieviele Körner hier liegen
k . . .	ersetze diese Landschaft durch die Kopie einer bereits vorhandenen anderen Landschaft
? . . .	zeige diese Erklärung
ESC . .	brich den Progammlauf ab

Die Richtungstasten (Pfeile) bedeuten:

Drehe Dich (Hamster) in die gewünschte Richtung. Falls Du schon diese Richtung hattest, gehe ein Feld weiter in diese Richtung.

2.3.3 Startposition des Hamsters, Backentaschen

Bei Beendigung des Landschaftseditors ("h") wird die aktuelle Hamsterposition und seine Richtung notiert. Die Anzahl der Körner, welche bei Programmstart in den Backentaschen vorhanden sein sollen, wird erfragt und ebenfalls notiert.

2.4 Ausführen von Hamsterprogrammen

Hamsterprogramme müssen in der Landschaft auf dem Bildschirm ablaufen. Daher muß vor Ausführen des ersten Hamsterbefehls die Landschaft auf den Bildschirm gebracht werden. Aus diesem Grunde werden Hamsterprogramme nicht mit 'run' sondern mit einem eigenen Aufruf des ELAN – Compilers 'lauf' gerufen. 'lauf' funktioniert genau so wie 'run', d.h. es benutzt auch 'last param'. Man kann also, wenn man soeben ein Hamsterprogramm mit dem EUMEL – Editor geschrieben oder korrigiert hat, den Hamstercompiler mit 'lauf' ohne Parameter starten. Man kann aber auch als Parameter den Namen der Datei, in der das auszuführende Hamsterprogramm steht, angeben.

2.4.1 Wirkung der Basis – Befehle und – Tests

'vor' bewirkt:

- ein Eingabezeichen wird von der Tastatur gelesen. Ausgewertet wird wie unter 2.4.4 beschrieben.
- es wird je nach Verzögerungsfaktor gewartet
- Falls die Kachel vor dem Hamster frei ist und noch zur Landschaft gehört, geht der Hamster um eine Kachel in seiner Richtung.
- Falls vor dem Hamster ein Hindernis liegt, oder wenn er im Begriff ist, aus der Landschaft hinauszulaufen, wird das Programm mit entsprechender Fehlermeldung abgebrochen.

'links um' bewirkt:

- wie bei 'vor' Annahme eines Tastendruckes und Warten.
- eine Drehung des Hamstes um 90 Grad gegen den Uhrzeigersinn. Das Zeichen, welches den Hamster und seine Richtung darstellt, ändert sich.

'nimm' bewirkt:

- wie bei 'vor' Annahme eines Tastendruckes und Warten.
- falls auf der Kachel, auf der der Hamster steht, kein Korn liegt, wird das Programm mit entsprechender Fehlermeldung abgebrochen.
- falls dort genau ein Korn liegt, wird dieses auf dem Bildschirm entfernt. Es wird zu denen in den Backentaschen addiert.
- falls mehrere Körner dort liegen, wird eines zu denen in den Backentaschen addiert, und von denen auf der Kachel subtrahiert.

'gib' bewirkt:

- wie bei 'vor' Annahme eines Tastendruckes und Warten.
- Falls die Backentaschen leer sind, wird das Programm mit entsprechender Fehlermeldung abgebrochen.
- Falls auf der Kachel schon ein Korn oder mehrere Körner liegen, wird zu ihnen eines addiert und von denen in den Backentaschen subtrahiert. Der Bildschirm ändert sich nicht.
- Falls noch kein Korn auf dieser Kachel liegt, wird eines auf den Bildschirm gezeichnet und von denen in den Backentaschen subtrahiert.

'vorn frei'

liefert den Wahrheitswert TRUE, wenn vor dem Hamster keine Hindernis-kachel liegt, also auch dann, wenn der Hamster im Begriff ist, über die Landschaft hinauszulaufen!

Wenn vor dem Hamster eine Hinderniskachel liegt, wird der Wahrheitwert FALSE geliefert.

'korn da'

liefert den Wahrheitswert TRUE, wenn auf der Kachel, auf der der Hamster streht, mindestens ein Korn liegt. Ansonsten wird der Wert FALSE geliefert.

'backen leer'

liefert den Wahrheitwert TRUE, wenn kein Korn in den Backentaschen notiert ist. Ansonsten FALSE.

2.4.2 Beschleunigen und Verzögern des Hamsterlaufs

Der Hamster hat vor Ausführung immer eine aktuelle Wartezeit. Bei Eingabe von '+' während des Hamsterlaufs wird diese Zeit halbiert (in gewissen Grenzen). Bei Eingabe von '–' wird sie bis zu einer bestimmten Grenze verdoppelt.

2.4.3 Programmabbruch durch Tastendruck

Kann zu jeder Zeit während der Landschaftserstellung und während eines Programmablaufs erreicht werden durch Tippen der Taste ESC. Dadurch ist es möglich, fehlerhafte Hamsterprogramme, welche in Endloswiederholungen münden, ohne großen Aufwand abzubrechen. Das gilt jedoch nur für solche Endloswiederholungen, welche mindestens einen ausgeführten Basis-Befehl haben. Denn nur dort wird (beim Verzögern) Tasteneingabe entgegengenommen.

Die Wiederholung:

```
WHILE korn da REPEAT
  IF vorn frei THEN vor END IF
END REPEAT
```

kann mit ESC dann nicht beendet werden, wenn der Hamster vor einem Hindernis auf einer Kachel steht, auf der mindestens ein Korn liegt. In diesem Fall hilft nur die globale Programmabbruchmöglichkeit SV und dann 'halt'.

2.4.4 Tasteneingabe und -auswertung

Während der Hamsterlaufs wirken folgende Tasten:

ESC	beende den Lauf
+ .	laufe schneller
– .	laufe langsamer
? .	zeige diese Erklärung (während des Programmablaufs!)

Alle anderen Tastendrücke können im Hamsterprogramm entgegengenommen werden durch die Prozedur 'taste':

TEXT PROC taste
liefert jeweils ein eingetipptes Zeichen. Die Taste ESC, welche zum Programmabbruch dient (siehe 2.4.3), wirkt sich schon aus, bevor sie hier als Wert geliefert werden könnte. Alle anderen Tasten werden als Wert geliefert, auch '+', '-' und '?'.

Damit können auch Hamsterprogramme geschrieben werden, in denen der Hamster auf Tasten reagiert (z. B. Spiele).

2.4.5 Angabe von Landschaftsnamen im Hamsterprogramm

Durch die Verwendung der Prozedur

```
PROC landschaft( TEXT CONST landschaftsname)
```

in einem Hamsterprogram kann der Dialog zu Beginn des Ablaufes eines Hamsterprogramms, welcher zum Erstellen der Landschaft dient, umgangen werden.

Wichtig ist dabei, daß die Prozedur 'landschaft' vor dem ersten Basis-Befehl stehen muß. Mit der Prozedur 'landschaft' ist es außerdem möglich, den Hamster in einem einzigen Programm durch mehrere Landschaften laufen zu lassen (z.B. für Spiele).

3. Installation des Hamstermodells

Das Paket 'hamsterbasis' wird in einer Task insertiert. Danach kann das Hamstermodell benutzt werden. Es bietet sich an, alle Tasks, in welchen der Hamster programmiert werden soll, als Söhne der Task anzulegen, in der das Paket 'hamsterbasis' insertiert ist.

Softwareklasse 2:

Regelmäßige Wartung

Autor:

Lothar Oppor

Kontaktadresse:

Gesellschaft für
Mathematik und Datenverarbeitung (GMD)
Schloß Birlinghoven
5202 St. Augustin

Umschlaggestaltung:

Hannelotte Wecken

Druck:

Zentrale Vervielfältigungsstelle der
Universität Bielefeld