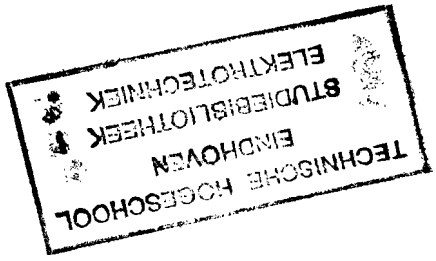
Een objectprocessor voor een interactief grafisch systeem

van de Meerendonk, M.M.L.

Award date:
1977

AFDELING DER ELEKTROTECHNIEK
TECHNISCHE HOGESCHOOL
EINDHOVEN
Groep Meten en Regelen



EEN OBJECTPROCESSOR VOOR EEN INTERACTIEF GRAFISCH SYSTEEM

door M.M.L. van de Meerendonk

Rapport van het afstudeerwerk
uitgevoerd van 1-9-'76 tot 27-10-'77
onder leiding van Prof.ir. F.J. Kylstra
ing. L.R.A. Kessener
ir. M. van der Woude

Samenvatting

In dit verslag is beschreven hoe drie-dimensionale geometrische informatie gestructureerd kan worden opgeslagen in een geheugen. Er zijn een aantal operaties gedefinieerd, waarmee met de informatie in zo'n geheugenstructuur kan worden gemanipuleerd.
Ook is een operatie ingevoerd, waarmee de opgeslagen informatie op een grafisch apparaat zichtbaar kan worden gemaakt.
Het geheel van operaties en datastructuur, waarop deze operaties van toepassing zijn, is objectprocessor genoemd.
Voor de besturing van deze objectprocessor is als voorbeeld gebruik gemaakt van een interactief programma.

INHOUDSOPGAVE

	blz.
1. Inleiding	5
2. Een interactief grafisch systeem	7
2.1. Weergave van drie-dimensionale geometrische informatie	8
2.2. Motivering voor het invoeren van de objectprocessor	13
3. De objectprocessor	14
3.1. Het begrip object	15
3.2. Operaties op objecten	16
3.3. Functies van de objectprocessor	19
4. De objectdefinitie	21
4.1. Element-keuze	22
4.2. Element-specificatie	25
5. Implementatie van de objectprocessor	28
5.1. Invoer van gegevens	29
5.2. Implementatie van het object	34
5.2.1. De elementen in een objectstructuur	34
5.2.2. De attributen van een element	36
5.2.3. De data-types	40
5.2.4. De objectadministratie	43
5.3. Implementatie van de operaties	44
5.3.1. Procedures t.b.v. basis-operaties	44
5.3.2. Het beheren van een aantal objecten	48
5.3.3. Het weergeven van een object	50
5.3.4. Het manipuleren met objecten	52
5.3.5. Opslag van een object	57
6. De besturing van de objectprocessor	59
6.1. Procedures voor informatie-invoer	60
6.1.1. Het inlezen van een objectnaam	60
6.1.2. De invoer van element-gegevens	60
6.1.3. Tmatrix-definitie	64
6.2. Besturings-commando's	66

blz.

7. Nabeschouwing 75
8. Literatuur-overzicht 77
   8.1. Literatuurlijst 77
   8.2. Inhouds-beschrijving van de literatuur 78
9. Appendix
   A. Beschrijving van enkele grafische pakketten
      A.1. Onderzoek in Nederland op het gebied van computer graphics
      A.2. GINO-F
      A.3. Siggraph's Graphics Standards Planning Committee
   B. Het programma

1. INLEIDING

Bepaalde gegevens kunnen door mensen veel sneller geïnterpreteerd worden als deze op grafische wijze worden gepresenteerd. Bij programma's, waarin dergelijke gegevens verwerkt moeten worden, is deze vorm van gegevens-presentatie daardoor aan te bevelen boven een presentatie met getallen en/of tekst. De geometrische structuur van de informatie is daarbij afhankelijk van het toepassingsgebied en elk applicatie-programma heeft weer een eigen manier om met de grafische informatie te manipuleren. Deze verscheidenheid in representatie en manipulatie is aanleiding geweest tot het definiëren en implementeren van de z.g. objectprocessor. Met de objectprocessor wordt de mogelijkheid gecreëerd om grafische informatie op een geheugenstructuur af te beelden en (eventueel op interactieve basis) operaties op de grafische gegevens uit te voeren.

In een interactief grafisch systeem is de objectprocessor een onderdeel, dat een hoeveelheid geometrische informatie beheert, waarmee het m.b.v. een aantal operaties kan manipuleren. Voor het weergeven van deze informatie op grafische apparaten maakt de objectprocessor gebruik van GINO-F subroutines.

De drie-dimensionale geometrische structuur, die in het geheugen kan worden afgebeeld, noemen we een object. Zo'n object is voor te stellen als een collectie van elementen. Als elementen hebben we gekozen voor punten, lijnstukken, contouren, tekst- en object-elementen. Bij de implementatie van de objectprocessor is een datastructuur ontwikkeld, waarin een object kan worden afgebeeld. De objectprocessor beheert meerdere objecten. Een objectadministratie zorgt er o.a. voor dat de verschillende objecten in het geheugen d.m.v. namen bereikbaar zijn.

Voor het transport van grafische informatie (element-gegevens) van en naar de objectprocessor zijn een aantal variabelen ingevoerd. Bepaalde procedures in de objectprocessor kunnen uit deze informatie een object opbouwen. Er zijn ook nog procedures t.b.v. het veranderen, het weergeven en het opslaan van objecten.

Voor de besturing van de objectprocessor is gekozen voor een interactief programma.

Het geheel is geschreven in de taal PASCAL.

2. EEN INTERACTIEF GRAFISCH SYSTEEM

De objectprocessor moet als onderdeel van een interactief grafisch systeem een aantal functies kunnen vervullen. Deze functies hebben betrekking op het manipuleren met grafische informatie, die in de vorm van objecten in het geheugen aanwezig is. De operaties, die in de objectprocessor voor deze functies zijn ingevoerd, zullen o.a. tot doel hebben: het opbouwen van een object in een geheugenstructuur en het aanbrengen van veranderingen in een object. M.b.t. deze operaties is het noodzakelijk, dat de objectprocessor kan worden voorzien van grafische informatie. Hiervoor bestaan een aantal apparaten zoals tablets, displays met lightpen, etc.

Een andere operatie, die de objectprocessor in een grafisch systeem moet kunnen uitvoeren, is het weergeven op een grafisch apparaat van de grafische informatie, waarover de objectprocessor in de vorm van een object beschikt. Voor deze uitvoer van gegevens bestaan eveneens speciale apparaten: refreshed/storage displays, plotters, etc.

Voor beschrijvingen van zowel de invoer- als de uitvoer apparaten verwijs ik naar [1].

Bij gebruik van bepaalde apparaten kan sprake zijn van een communicatie tussen gebruiker en systeem. In zo'n geval spreken we van een interactief grafisch systeem. Het gevolg van een operatie, die interactief wordt uitgevoerd, is b.v. dat de door deze operatie aangebrachte verandering in een object direct resulteert in een overeenkomstige verandering in de afbeelding, die van dat object op een apparaat (b.v. refreshed display) aanwezig is.

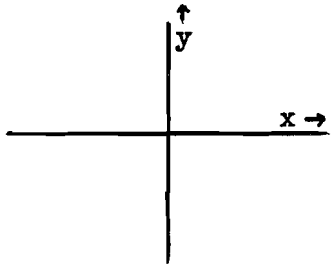
In de inleiding is reeds vermeld, dat de objectprocessor voor de weergave gebruik maakt van het subroutine-pakket GINO-F. Door deze vorm van uitvoer is de objectprocessor device-onafhankelijk te noemen. M.a.w. de objectprocessor genereert uitvoer voor een virtuele tekenmachine.
Op welke wijze de drie-dimensionale informatie in een object door deze tekenmachine op een twee-dimensionaal scherm wordt afgebeeld, is in de volgende paragraaf beschreven.

2.1. Weergave van drie-dimensionale geometrische informatie (Viewing transformation)

We willen een object, dat gedefinieerd is in een drie-dimensionaal coördinaten-stelsel, afbeelden op een twee-dimensionaal vlak. Voor de projectie, die hierbij noodzakelijk is, definiëren we in het coördinaten-stelsel van het object:

- een punt V, van waaruit het object bekeken wordt (viewing-point)
- een punt D, waarmee de kijkrichting (viewing-direction: van V naar D) en de positie van het projectie-vlak is bepaald (zie fig. 2.1).

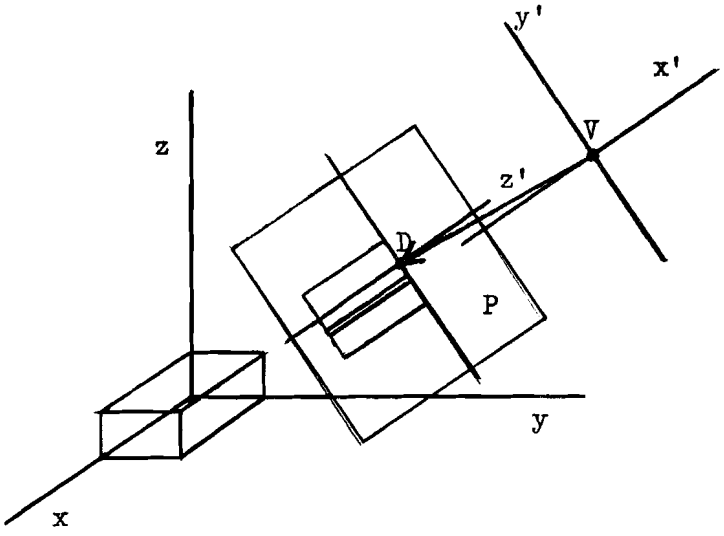
Er wordt nu een drie-dimensionaal coördinaten-stelsel gedefinieerd, waarvan de oorsprong het viewing-point V is en de positieve z-as door het punt D gaat. De viewing-direction is hierin dus richting positieve z-as gedefinieerd. Het projectie-vlak ligt loodrecht op de z-as door het punt D. Wordt het x-y-vlak van dit coördinaten-stelsel op het projectie-vlak geprojecteerd, dan moet het afgebeelde assenstelsel er als volgt uitzien:



Dit is in overeenstemming met de coördinaten-stelsels van de meeste grafische uitvoer-apparaten.

Er is nu nog een vrijheidsgraad voor het nieuwe coördinaten-stelsel en wel de oriëntatie van de x- en de y-as. We kiezen hiervoor de x-as van het nieuwe coördinaten-stelsel evenwijdig aan het x-y-vlak van het object-coördinaten-stelsel.

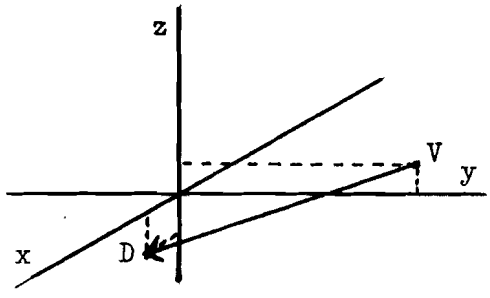
Het object, zoals het in het oorspronkelijke coördinaten-stelsel is gedefinieerd, moet nu worden getransformeerd naar het nieuwe coördinaten-stelsel. Dit gebeurt d.m.v. een aantal lineaire trans-



Figuur 2.1. Projectie van een object op een vlak P.

formaties, die in onderstaande volgorde op het object moeten worden uitgevoerd:

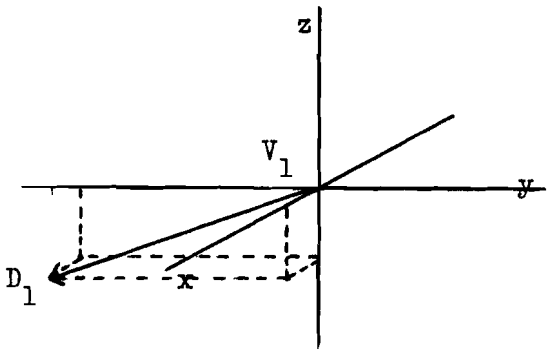
— Originele toestand.
  V : viewing-point
  V ↔ D : viewing-direction



Figuur 2.2a. Viewing-point en viewing-direction in het object-coördinaten-stelsel.

— Translatie : viewing-point in oorsprong.

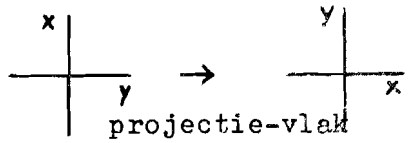
$$T_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -V_x & -V_y & -V_z & 1 \end{pmatrix}$$

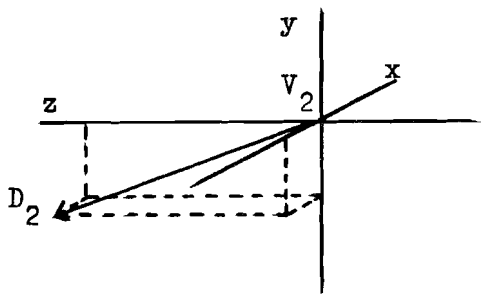


Figuur 2.2b. Translatie.

— Permutatie van assen :

$$T_2 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

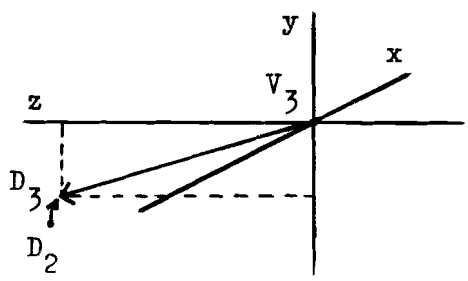




Figuur 2.2c. Permutatie van assen.

— Rotatie om y-as : vector $V_3 \rightarrow D_3$ in y-z-vlak.

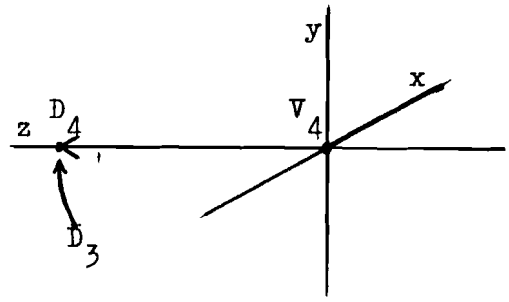
$$T_3 = \begin{pmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Figuur 2.2d. Rotatie om y-as.

— Rotatie om x-as : D_4 op z-as.

$$T_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) & 0 \\ 0 & \sin(\theta_x) & \cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Figuur 2.2e. Rotatie om x-as.

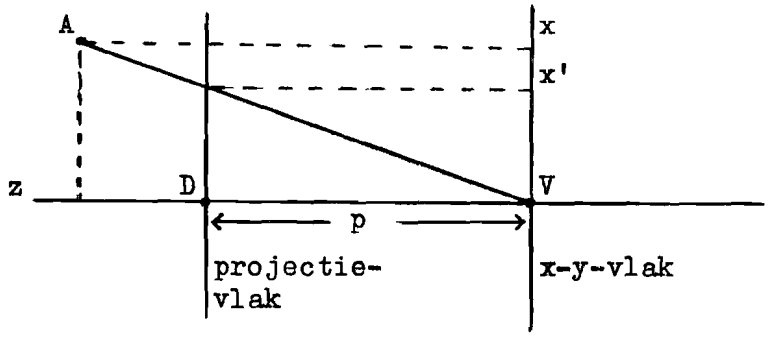
Door te werken met homogene coördinaten (zie [1]) kan deze afbeelding uitgevoerd worden d.m.v. vermenigvuldiging van alle coördinaten met een (4 x 4)-matrix M. Deze matrix is gedefinieerd als:

$$M = T_1 . T_2 . T_3 . T_4 \quad \text{(zie fig. 2.2a t/m 2.2e).}$$

Voor een vector $\underline{x}$ in het originele coördinaten-stelsel geldt dan dat de corresponderende vector $\underline{x}'$ in het nieuwe coördinaten-stelsel gelijk is aan:

$$(x', y', z', 1) = (x, y, z, 1) * M.$$

Is de drie-dimensionale geometrische informatie met de matrix M vermenigvuldigd, dan is de volgende stap het projecteren van de nieuwe drie-dimensionale coördinaten op het projectie-vlak P.
Bij een axonometrische projectie zijn door de matrix-vermenigvuldiging meteen de x- en y-coördinaten beschikbaar, die voor de twee-dimensionale weergave nodig zijn.
Bij een perspectieve projectie is nog een berekening nodig om de uiteindelijke beeldscherm-coördinaten te krijgen. Deze berekening wordt met onderstaande figuur duidelijk gemaakt (fig. 2.3).



Figuur 2.3. Perspectieve projectie van punt A.

Op het projectie-vlak is de x-coördinaat van de afbeelding van punt A (x, y, z) gedefinieerd als

$$x' = x * p/z \quad \text{met} \quad p = z_D - z_V.$$

Door het toepassen van de mogelijkheden, die homogene coördinaten bieden, is deze projectie onder te brengen in de viewing-transformation-matrix. Details hierover zijn te vinden in [2].

In het pakket GINO-F is nog een variatie op deze projectie ingevoerd met de subroutine FROM3 (zie GINO-F user manual).

Voor een literatuur-beschrijving van deze viewing transformation verwijs ik naar [1], waarin ook hier nog niet genoemde begrippen als window, viewport, etc. aan de orde komen.

2.2. Motivering voor het invoeren van de objectprocessor

In de inleiding van dit hoofdstuk is gesproken over een interactief grafisch systeem, waarmee het mogelijk moet zijn om:

- drie-dimensionale geometrische informatie af te beelden op een geheugenstructuur
- te manipuleren met de in het geheugen afgebeelde objecten
- de drie-dimensionale geometrische informatie, die in het geheugen is ondergebracht, op een twee-dimensionaal scherm zichtbaar te maken.

In het programma wordt bij de laatste operatie gebruik gemaakt van de subroutines van het pakket GINO-F.

De vraag is nu hoe in het programma de opslag van de grafische informatie plaatsvindt en op welke wijze operaties hierop worden uitgevoerd. Mogelijkheden om hier eveneens gebruik te maken van bestaande subroutine-pakketten zijn er niet (zie appendix A: beschrijving van enkele subroutine-pakketten). Een pakket als GINO-F b.v. kent wel subroutines voor o.a.

- het tekenen van geometrische informatie,
- het selecteren van devices,
- het uitvoeren van (viewing) transformaties,

maar een middel om gestructureerde afbeeldingen van drie-dimensionale figuren in een geheugen te creëren is niet beschikbaar. Manipuleren met de grafische informatie, zoals het hier gewenst is, is dus niet mogelijk. Daarom is besloten tot het bouwen van de objectprocessor. Hiervoor is het begrip object ingevoerd en zijn operaties gedefinieerd, die hierop kunnen worden uitgevoerd. Bij de implementatie is een datastructuur gedefinieerd, waarin objecten kunnen worden gerepresenteerd. Voor het uitvoeren van de operaties zijn een aantal procedures geschreven.

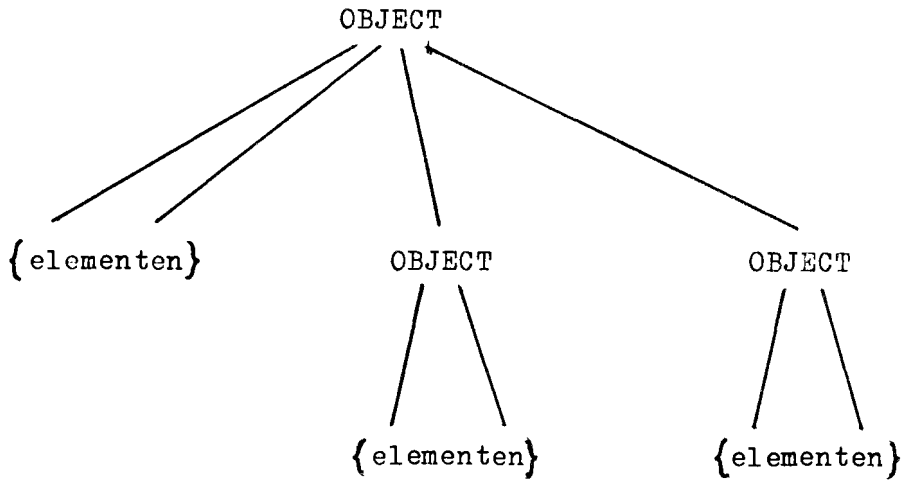
3. DE OBJECTPROCESSOR

In hoofdstuk 2 is een object gedefinieerd als de beschrijving van een drie-dimensionale structuur. De objectprocessor is beschreven als een verzameling van procedures, die kunnen opereren op de objecten, die afgebeeld zijn in een datastructuur.

In dit hoofdstuk wordt een meer gedetailleerde beschrijving van de objectprocessor gegeven. Dit begint in par. 3.1 met de opbouw van een object. Een opsomming van de operaties, die op objecten moeten kunnen worden uitgevoerd, is in par. 3.2 gegeven. In par. 3.3 komen de functies aan de orde, die de objectprocessor bezit voor het realiseren van de objectoperaties.

3.1. Het begrip object

We definiëren een object als een collectie van elementen. Het aantal elementen in een object is onbeperkt en bovendien is het toegestaan, dat in een object identieke elementen aanwezig zijn. Elk element bevat een hoeveelheid geometrische informatie en informatie, die bij weergave de representatie bepalen, de z.g. attributen. We onderscheiden enkelvoudige en samengestelde elementen. Enkelvoudige elementen kunnen zijn: punten (één coördinaten triple), lijnstukken (twee coördinaten triples), etc. De samengestelde elementen zijn gedefinieerd als een collectie van elementen, een object dus eigenlijk. Met deze beschrijving is een object-structuur ontstaan, die in figuur 3.1 wordt weergegeven.



Figuur 3.1. De structuur van een object.

Bij de definitie van objecten is geen recursie toegestaan. Een object heeft daardoor de structuur van een gerichte acyclische graaf.

In hoofdstuk 4 wordt de beschrijving van een object nader uitgewerkt met o.a. een specificatie van de verschillende elementen

3.2. Operaties op objecten

Voor het manipuleren met objecten zijn de volgende operaties ingevoerd:

- Het toevoegen van een element aan een object:
  ADDELEMENT (obj, e).
  Het object obj wordt met het element e uitgebreid.

- Het verwijderen van een element uit een object:
  DELELEMENT (obj, e).
  Indien aanwezig wordt het element e uit het object obj verwijderd.
  Het aanbrengen van wijzigingen in een element is mogelijk door het desbetreffende element eerst uit het object te verwijderen om daarna het juiste element toe te voegen.

- Het transformeren van een object:
  TRANSFOBJECT (obj1, obj2, tmatrix).
  Deze operatie heeft uitsluitend betrekking op lineaire transformaties: scale, translate, rotate, shear.
  Met deze operatie wordt een object, dat in een drie-dimensionaal coördinatenstelsel is gedefinieerd, d.m.v. een aantal lineaire transformaties in een nieuwe drie-dimensionale ruimte afgebeeld.

- Vereniging van twee objecten:
  UNION (obj1, obj2, obj3).
  Door deze operatie wordt een object (obj3) gecreëerd, dat bestaat uit de elementen van zowel object obj1 als van object obj2. Komt een element e meermalen voor in beide objecten (m.a.w. zijn er in die objecten meerdere elementen, die wat betreft geometrische informatie en attributen identiek zijn), dan geldt voor het aantal elementen e in object obj3:

  $$(\#e \text{ in } obj3) = (\#e \text{ in } obj1) + (\#e \text{ in } obj2)$$

  Opm.: Met #e wordt het aantal elementen aangegeven, die voldoen aan de beschrijving van e.

Er geldt zodoende altijd:

(#elem.n in obj3) = (#elem.n in obj1) + (#elem.n in obj2)

Het is bij deze operatie toegestaan om identieke objectnamen te gebruiken, b.v.

UNION (obj1, obj1, obj2).

Dit geldt ook voor de operaties TRANSFOBJECT, DIFF en INTERS.

- Verschil van twee objecten:
  DIFF (obj1, obj2, obj3).
  Deze operatie levert een object (obj3) af, dat is gedefinieerd uit de elementen, die wel in object obj1 maar niet in object obj2 voorkomen. Komt het voor dat een bepaald element e meermalen aanwezig is in beide objecten, dan wordt het uiteindelijke aantal van dat element e in het nieuwe object:

  (#e in obj3) = (#e in obj1) - (#e in obj2).

  Hierdoor geldt dus altijd:

  (#elem.n in obj3) = (#elem.n in obj1) - (#elem.n in obj2).

- Doorsnede van twee objecten:
  INTERS (obj1, obj2, obj3).
  Er ontstaat door deze operatie een object, dat is gedefinieerd uit die elementen, die zowel in object obj1 als in object obj2 voorkomen. Dit nieuwe object krijgt de naam obj3. Komt het voor dat een bepaald element e meermalen aanwezig is in beide objecten, dan wordt het uiteindelijke aantal van dat element e in het object obj3:

  (#e in obj3) = min{(#e in obj1),(#e in obj2)}.

  De operatie is mogelijk door tweemaal achtereen de operatie DIFF uit te voeren:

  DIFF (obj1, obj2, obj3)
  DIFF (obj1, obj3, obj3).

- Alle elementen verwijderen uit een object:
  EMPTY (obj).
  Na deze operatie is het aantal elementen in object obj gelijk aan nul.

- Box-operatie:
  BOX (objl, obj2, box-definitie).
  In het drie-dimensionale coördinaten-stelsel van object objl wordt een rechthoekige doos gedefinieerd. Object obj2 bestaat na de operatie uit:

  - die elementen van object objl, die geheel binnen die doos liggen,
  - en de elementen, die na clipping van de gedeeltelijk binnen de doos aanwezige elementen ontstaan.

- Een object weergeven:
  DISPLAY (obj, V, D, window, viewport).
  In par. 2.1 is beschreven hoe de viewing transformation bij deze operatie plaatsvindt. M.b.v. viewing point (V) en kijkrichting (V → D) wordt bepaald welk aanzicht van het object op het scherm moet verschijnen. Met de window kan aangegeven worden welk gedeelte van het geprojecteerde beeld op het scherm zichtbaar moet worden. Een viewport kan gebruikt worden om een bepaald gedeelte van het scherm aan te geven, waarop de inhoud van de window getekend moet worden. Bijzonderheden zoals hidden line elimination, perspectief, etc. (indien beschikbaar) kunnen worden gebruikt om de afbeelding zo realistisch mogelijk te maken.

3.3. Functies van de objectprocessor

De functies, die de objectprocessor moet kunnen vervullen, zijn:

- Het beheren van een aantal objecten.
  Operaties, die hiervoor zijn gedefinieerd, hebben voornamelijk betrekking op het bijhouden van een object-administratie.
  Hiertoe behoort o.a. het declareren van een object (NEWOBJECT): voordat een operatie op een object kan worden uitgevoerd, zal eerst de naam aan de objectprocessor moeten worden opgegeven, waaronder het object bereikbaar moet worden.
  Ook een operatie voor het verwijderen van een object valt hieronder (REMOBJECT).

- Het weergeven van een object.
  De operatie, die voor het weergeven van een object noodzakelijk is, is in par. 3.2 aan de orde geweest (DISPLAY).

- Het manipuleren met objecten.
  De operaties, die hierop betrekking hebben, zijn in par. 3.2 opgenoemd: ADDELEMENT,
  DELELEMENT,
  TRANSFOBJECT,
  UNION,
  DIFF,
  INTERS,
  EMPTY en
  BOX.
  Een operatie, die geen verandering in een object veroorzaakt en daarom nog niet is vermeld, is MEMBER (obj, e).
  Hiermee wordt alleen gecontroleerd op de aanwezigheid van een element e in het object obj.

- Het opslaan van objecten.
  Objecten moeten in een file op een achtergrondgeheugen (b.v. pack) kunnen worden opgeslagen, opdat ze in een later stadium weer beschikbaar zijn.

Operaties hiervoor zijn:

- STOREOBJECT
  een object opslaan in een file.
- GETOBJECT
  een object uit een file in het geheugen plaatsen.

We kunnen dit hoofdstuk besluiten met de volgende voorstelling van de objectprocessor:

De objectprocessor kan worden beschouwd als een black box, die een aantal besturingsmogelijkheden bezit.
D.m.v. deze besturing is het mogelijk operaties uit te voeren op de objecten, die in de black box aanwezig zijn.
Geometrische informatie wordt in de vorm van element-beschrijvingen aan de black box afgegeven en de uitvoer bestaat uit objecten, die m.b.v. GINO-F op een fysisch apparaat kunnen worden afgebeeld.

4. DE OBJECT-DEFINITIE

Bij het opstellen van een object-definitie moet met het volgende rekening worden gehouden:

1 - Een geometrische figuur moet op een eenvoudige manier kunnen worden afgebeeld op een object.
2 - Een object moet zo kunnen worden afgebeeld op een geheugenstructuur, dat de verschillende operaties optimaal kunnen worden uitgevoerd.

In dit hoofdstuk wordt een object nu zodanig bepaald, dat aan de eerste voorwaarde voldaan is. De tweede eis zal in hoofdstuk 5 bij de implementatie aan de orde komen.

4.1. Element-keuze

In par. 3.1 is gesteld, dat een object is opgebouwd uit verschillende elementen. Nu worden de elementen bepaald, waarmee een geometrische figuur in een object kan worden afgebeeld.
Bij een drie-dimensionaal grafisch systeem bestaat er in eerste instantie aanleiding om bij het kiezen die elementen als kandidaten te nemen, die representatief zijn voor een bepaalde dimensie:

- punten (x,y,z)
- lijnen (1-dim.) :
  - rechte : $(x_1,y_1,z_1) + \lambda(x_2,y_2,z_2)$
  - lijnstuk : $(x_b,y_b,z_b),(x_e,y_e,z_e)$
- vlakken (2-dim.) :
  - onbegrensd vlak :
    $(x_1,y_1,z_1) + \lambda(x_2,y_2,z_2) + \mu(x_3,y_3,z_3)$
  - gesloten vlakken, bepaald door een contour met n hoekpunten:
    $(x_1,y_1,z_1),(x_2,y_2,z_2), \ldots ,(x_n,y_n,z_n)$
- lichamen (3-dim.) : tetraëder.

Van deze elementen kunnen onbegrensde vlakken moeilijk twee-dimensionaal worden weergeven. Hierdoor is, mede door het feit dat een object in het algemeen toch een afbeelding is van een of andere realiteit, besloten om oneindige vlakken als element-type niet in te voeren.

In tegenstelling tot oneindige vlakken is het wel mogelijk een rechte twee-dimensionaal weer te geven. Het is niet noodzakelijk hier een apart element voor te definiëren. Een rechte is in een object namelijk onder te brengen in de vorm van een element-definitie van een lijnstuk met een grote lengte.

Ofschoon lichamen in eerste instantie als elementen zijn voorgesteld, is het mogelijk om ze in een object te representeren d.m.v. een aantal vlakken, die de door zo'n lichaam ingenomen ruimte precies omsluiten. Op deze manier wordt wel niet aangegeven, dat het om de inhoud te doen is , maar in een twee-dimensionale tekening

is dit toch niet duidelijk te maken. We zullen daarom een lichaam (tetraëder) niet als element invoeren.

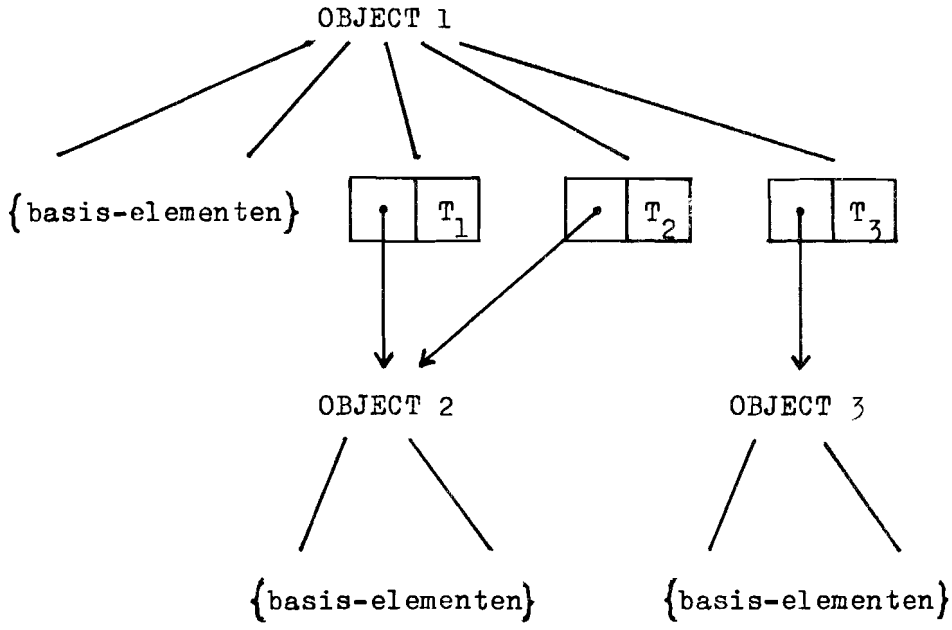
Of het mogelijk is om uit bovengenoemde elementen voor driedimensionale geometrische informatie een object-definitie samen te stellen, wordt nu onderzocht met andere elementaire delen, die in een geometrische figuur kunnen voorkomen.

- krommen (functies zoals: $x^2+y^2=c$, $z=0$)
  We kunnen een kromme benaderen d.m.v. een aaneenschakeling van korte lijnstukjes, waarbij de lengte van elk afzonderlijk lijnstukje afhankelijk is van de op die positie optredende richtings-verandering in de kromme.
- gewelfde vlakken (functies zoals $x^2+y^2+z^2=c$)
  Het probleem bestaat hier niet alleen uit het voor opslag geschikt maken van de vlak-informatie, maar ook nog uit de vraag, hoe we zo'n vlak op het scherm duidelijk moeten weergeven. De oplossing voor beide problemen is de methode, waarbij zo'n oppervlak wordt voorgesteld door een aaneengesloten geheel van een groot aantal oppervlakte-elementjes (b.v. driehoeken, rechthoeken, of vijfhoeken).
- tekst
  Net zoals krommen zijn ook characters op te bouwen uit een aantal korte lijnstukjes, die dan wel in één vlak moeten liggen. Het is dus mogelijk om characters in de vorm van een aantal lijnstukjes in een object onder te brengen. Omdat bij bepaalde devices de mogelijkheid bestaat characters m.b.v. hardware te genereren, is als element ook de text-string ingevoerd.

Met het text-element erbij zijn we nu gekomen tot een totaal van vier zogenaamde basis-elementen:

- punt
- lijnstuk
- contour
- text.

Als vijfde en tevens laatste element van een object introduceren we het object zelf. Door aan zo'n object-element nog een transformatie-matrix (lineaire transformaties!) te koppelen wordt het mogelijk, reeds gedefinieerde objecten op een bepaalde manier, eventueel met enkele basis-elementen, samen te voegen tot een nieuw object.



Figuur 4.1. Een object bestaande uit basis-elementen en object-elementen.

Gewenst is hierbij, dat een object meermalen (verschillend getransformeerd) als object-element in een object kan voorkomen (zie fig. 4.1). Een object-element is dus een element, dat bestaat uit een verwijzing naar een object en een matrix, die de transformatie op dit object beschrijft. Deze transformatie kan beschouwd worden als een opeenvolging van lineaire transformaties als scale, translate, rotate en shear (vermenigvuldiging van matrices).

Door de invoering van het object-element heeft een object de structuur gekregen van een gerichte graaf, waarin echter door de noodzaak van een eindige object-definitie geen cycles mogen voorkomen.

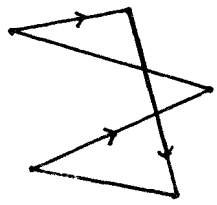
4.2. Element-specificatie

Nu we ons een object kunnen voorstellen als een collectie van punten, lijnstukken, contouren, tekst- en object-elementen kunnen we de informatie gaan bepalen, die voor elk type element van belang is. We zullen daarbij onderscheid maken tussen geometrische informatie en attributen. Hieronder volgt nu voor elk type element een opsomming van de informatie, waarmee een element van dat type gedefinieerd is.

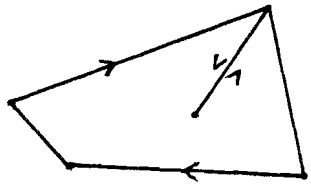
- Punt:
  - één coördinaten triple
  - attributen:
    pendikte, penkleur

- Lijnstuk:
  - een coördinaten triple voor het beginpunt
    een coördinaten triple voor het eindpunt
    eis: beginpunt $\neq$ eindpunt (lengte $\neq 0$)
  - attributen:
    pendikte, penkleur, lijnrepresentatie

- Contour:
  - minstens drie niet-identieke coördinaten triples, die gedefinieerd zijn in een volgorde, die bepalend is voor de normaal van het vlak (oppervlakte $\neq 0$). Vereist is hierbij nog, dat alle coördinaten triples in één vlak moeten liggen en dat de verbindingslijnen tussen de achtereenvolgende hoekpunten (omtrek) elkaar niet snijden.
    v.b.



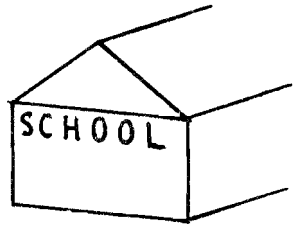
verboden



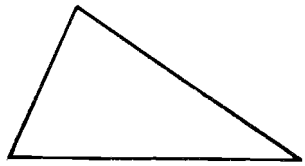
toegestaan

- attributen:
  pendikte, penkleur, lijn-representatie, vlak-representatie

- Tekst:
  Een text-string kan in een object op verschillende manieren als element voorkomen, b.v.
  - als onderdeel van een object:
    de text-string neemt in het object een vaste positie in t.o.v. de andere elementen.
    b.v. tekst op een muur



  - als commentaar:
    de positie van de text-string wordt in het object-coördinaten-stelsel door één punt bepaald. De afbeelding van dit punt op een grafisch apparaat is het links-onder-punt van de text-string, die hierop horizontaal wordt weergegeven.
    b.v.



.driehoek

Om aan te geven hoe de text-string nu moet worden geïnterpreteerd, is het attribuut 'quality' ingevoerd.
Voor de positie van de text-string heeft het eerste voorbeeld de meeste informatie nodig, n.l.

- links-onder-punt van de text-string (P_{lo})
- de schrijfrichting ($P_{lo} \longrightarrow P_{ro}$)
- de vlak-definitie (P_{lb})

Voor het tweede voorbeeld is daarentegen maar één coördinaten triple vereist, n.l. het links-onder-punt van de string (P_{lo}). Om aan beide gevallen te voldoen, wordt een definitie van een tekst-element gegeven, waarin de volgende informatie kan worden ondergebracht:

- de positie van de text-string, bepaald door:
  een coördinaten triple voor het links-onder-punt (P_{lo}),
  een coördinaten triple voor de schrijfrichting (P_{ro}),
  een coördinaten triple voor de definitie van het schrijfvlak (P_{lb})
- de characters van de text-string
- attributen:
  pendikte, penkleur, italicity, character-hoogte, character-breedte, quality

- Object-element:
  - een verwijzing naar een al bestaand object (geen cycles!)
  - een matrix, die het gerefereerde object kan transformeren naar de voorstelling, die het in het totale object moet innemen
  - attributen:
    Door het toekennen van attributen aan een object-element wordt het mogelijk de attributen, die behoren tot de elementen van het gerefereerde object, te beïnvloeden.
    Hiervoor zijn de begrippen relatieve en absolute attributen ingevoerd. Attributen, die als absoluut gedefinieerd zijn, zullen bij weergave onveranderd op hun element worden toegepast, terwijl relatieve attributen beïnvloedbaar zijn door de attributen van de object-elementen.
    Mogelijke attributen zijn:
    pendikte, penkleur, lijn-representatie, vlak-representatie

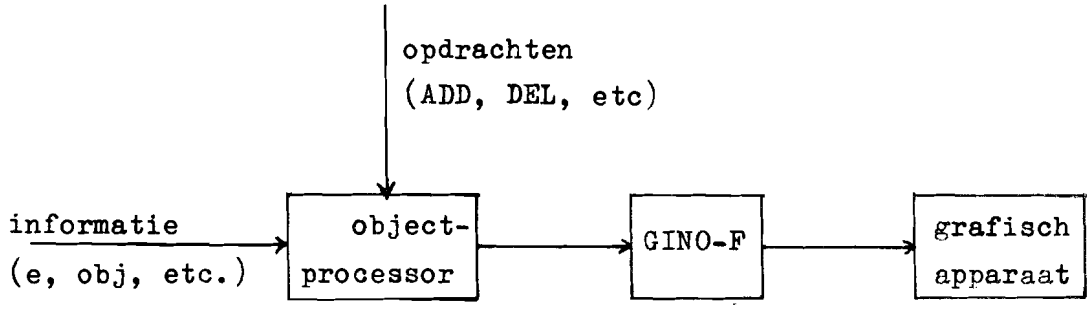
5. IMPLEMENTATIE VAN DE OBJECTPROCESSOR

De objectprocessor beheert een aantal objecten en bezit een aantal operaties, waarmee met deze objecten kan worden gemanipuleerd. Een opdracht aan de objectprocessor voor het uitvoeren van een bepaalde operatie zal aanleiding geven tot het aanroepen van procedures, die deze operatie kunnen realiseren. Het schrijven van alle procedures is een onderdeel geweest van de implementatie van de objectprocessor. De implementatie betekende ook het definiëren van een datastructuur, waarin een object kan worden afgebeeld.

Voor de implementatie is gebruik gemaakt van de taal PASCAL. Er is hiermee een data-type 'element' (record) gedefinieerd, dat voor de opslag van de geometrische informatie van een element kan worden gebruikt. Bij het opbouwen van een object-datastructuur, bestaande uit een aantal van die records, moet de objectprocessor worden voorzien van deze geometrische informatie. Er is voor deze gegevens-invoer een variabele 'e' van het type 'eleminf' (record) in de objectprocessor gedeclareerd, die een element-beschrijving en nog enkele andere belangrijke gegevens kan bevatten. Moet de informatie in 'e' in een record van het type 'element' worden ondergebracht, dan wordt dynamisch geheugenruimte voor het element opgeëist. M.b.v. een pointer (var. 'ep') naar deze geheugenruimte blijft een element binnen de objectprocessor bereikbaar. Pointers worden in procedures ook gebruikt om de locatie van een object aan te geven (var. 'op'). Er is een objectadministratie ingevoerd, waarin een pointer naar een object kan worden geregistreerd onder de naam, waaronder het object buiten de objectprocessor bekend is.

5.1. Invoer van gegevens

Een opdracht aan de objectprocessor zal resulteren in het aanroepen van enkele procedures. Deze procedures hebben informatie nodig, zoals: element-beschrijving, objectnaam, objectlocatie, transformatie-matrix. Een opdracht moet dus vergezeld gaan van een aantal van deze gegevens (fig. 5.1.).



Figuur 5.1. De besturing van de objectprocessor.

Er zijn in de objectprocessor een aantal variabelen gedeclareerd, waarin o.a. de naam van een 'current object' en de beschrijving van een 'current element' te vinden zijn:

var	type	informatie
e	eleminf	element-beschrijving
obj	array[1..10]of char	object-naam
op, op_1, op_2	object-pointer	locatie van object
i	integer	index van een objadm-entry
tm	(4 x 4) matrix	transformatie-matrix

'obj', 'op' en 'i' geven resp. de naam, de locatie en de objadm-entry van het current object weer. In verband met operaties, die op meer dan een object betrekking hebben, zijn de hulp-pointers 'op_1' en 'op_2' ingevoerd. De variabele 'e' bevat de beschrijving

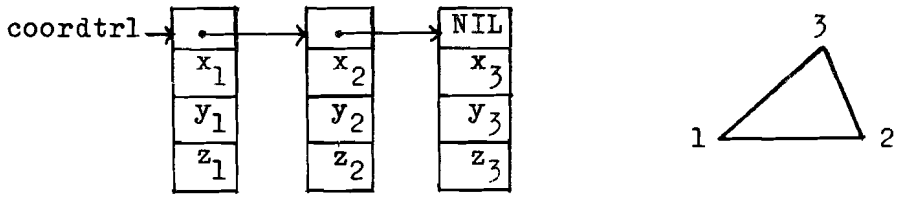
van het current element en evt. een pointer naar een element in de datastructuur, dat aan deze beschrijving voldoet (e.lastref).
De definitie van het type 'eleminf' luidt:

```
eleminf = record
            lastref : ↑element;
            obj : array[1..10] of char;
            relpen, relcol, rellst, relgrey : boolean;
            detect : boolean;
            pen, colour : -100..100;
            case elementtype : elem of
            point   : (pval : set of pointit;
                       x, y, z : real);
            line    : (lval : set of lineit;
                       xb, yb, zb, xe, ye, ze : real;
                       llst : ldescr);
            contour : (cval : set of contourit;
                       coordtrl : ↑coordtr;
                       nofcoordtrs : integer;
                       clst : ldescr;
                       cgrey : -100..100);
            text    : (tval : set of textit;
                       xlo, ylo, zlo, xro, yro, zro,
                       xlb, ylb, zlb : real;
                       chars : array[1..10] of char;
                       italic, charw, charh : real);
            objel   : (oval : set of objit;
                       objref : array[1..10] of char;
                       tmatrix : matrix;
                       olst : ldescr;
                       ogrey : -100..100)
          end; (*eleminf*)
```

Ontvangt de objectprocessor een element-beschrijving, dan wordt deze in de variabele 'e' geplaatst. Voor een volledige element-

beschrijving zullen de volgende items in 'e' gedefinieerd moeten zijn:

- element-type point:
  x, y, z
- element-type line:
  xb, yb, zb : beginpunt
  xe, ye, ze : eindpunt
- element-type contour:
  nofcoordtrs : aantal hoekpunten van de contour ($\geqslant 3$)
  alle hoekpunten van de contour :
    Elk hoekpunt wordt geplaatst in een record van het type 'coordtr'
    coordtr = record
              flink : ↑coordtr;
              x, y, z : real
              end; (*coordtr*).
    Het item coordtrl is een pointer naar het eerste coordtr-record in een kettingstructuur, waarin in een bepaalde volgorde (definitie van de normaal van het vlak!) de coördinaten-triples van de hoekpunten zijn afgebeeld (fig. 5.2).



Figuur 5.2. Opslag van de hoekpunten van een contour.

- element-type text:
  xlo, ylo, zlo : links-onder-punt van de text-string
  xro, yro, zro : schrijfrichting (afh. van text-presentatie)
  xlb, ylb, zlb : dit punt bepaalt samen met l.o.-punt en schrijfrichting het vlak, waarin de tekst geschreven moet worden (afh. van text-presentatie)
  chars : de text-string wordt geplaatst in een array van tien characters

- element-type objel:
  objref : de naam van het object, waarnaar verwezen wordt
  (controle op evt. cycles noodzakelijk!)
  tmatrix : de matrix, die de transformatie op het gerefereerde
  object beschrijft

Welke gegevens bij de invoer van een element zijn gedefinieerd en vervolgens in de daarvoor aanwezige items van de variabele 'e' zijn geplaatst, is geregistreerd in het item pval, lval, cval, tval of oval (afhankelijk van het type element). Dit zijn sets van resp.

```
pointit = (x, y, z, pdet, ppen, pcol);
lineit = (xb, yb, zb, xe, ye, ze, ldet, lpen, lcol, llst);
contourit = (coordtrs, cdet, cpen, ccol, clst, cgrey);
textit = (xlo, ylo, zlo, xro, yro, zro, xlb, ylb, zlb, chars,
          tdet, tpen, tcol, italic, charw, charh, qual);
objit = (objref, tmatrix, odet, open, ocol, olst, ogrey);
```

M.b.v. zo'n set van element-items is controle op de invoer van een volledige element-definitie mogelijk. Of bij een element attributen zijn gedefinieerd, is hierin ook geregistreerd. De attribuut-waarden zijn dan in 'e' te vinden in de items:

- detect : deze boolean geeft aan of het betrokken element detectable is (b.v. wordt bij het aanwijzen met een lightpen het elem. gesel.)
- pen : pendikte
- colour : penkleur
- llst, clst, olst : lijn-representatie (line-style)
- cgrey, ogrey : vlak-representatie (grey-scale of helderheid)
- italic : italicity
- charw : character-breedte
- charh : character-hoogte

Of de attributen geïnterpreteerd moeten worden als absolute of relatieve attributen, is aangegeven met de booleans relpen, relcol, rellst en relgrey. In par. 5.2.2 worden de attributen nader beschreven.

Opmerking m.b.t. het current element:

Is de pointer e.lastref≠NIL en maakt het element e.lastref deel uit van het object e.obj, dan bevat 'e' de beschrijving van dat element. Dit element is dan het laatste element geweest, dat bij de operaties ADDELEMENT of MEMBER betrokken was.
De aanwezigheid van een current element betekent, dat bij een volgende operatie m.b.t. dat element geen nieuwe element-beschrijving hoeft te worden gegeven.

Opmerking m.b.t. het current object:

Het gebruik van het current object, waarvan 'obj' de naam, 'op' de locatie en 'i' de index van de objadm-entry bevat, maakt het bij achtereenvolgende operaties op een bepaald object overbodig steeds opnieuw de naam van dat object te vermelden.

5.2. Implementatie van het object

5.2.1. De elementen in een objectstructuur

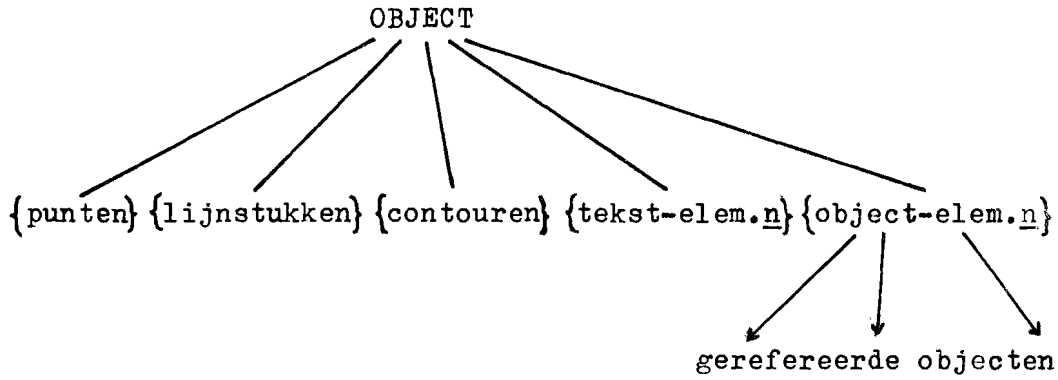
In hoofdstuk 4 is een object beschreven als een collectie van elementen. Er moeten een aantal operaties kunnen worden uitgevoerd op de elementen van een object. Deze operaties zijn gebaseerd op een aantal basis-operaties:

1 - Het aflopen van alle elementen in een object.
    We onderscheiden hierin:
    - het bereiken van alleen die elementen, die direct deel uit maken van het betrokken object (één niveau)
      b.v. TRANSFOBJECT
    - het bereiken van zowel de elementen, die direct, als de elementen, die indirect (via object-elementen) van het betrokken object deel uit maken (meerdere niveau's)
      b.v. DISPLAY
2 - Het verwijderen van een element uit een collectie.
    b.v. DELELEMENT, DIFF
3 - Het toevoegen van een element aan een collectie.
    b.v. ADDELEMENT, UNION
4 - Het selecteren van een element in een object.
    b.v. MEMBER, DIFF

De object-datastructuur wordt nu in een aantal stappen bepaald aan de hand van deze basis-operaties.

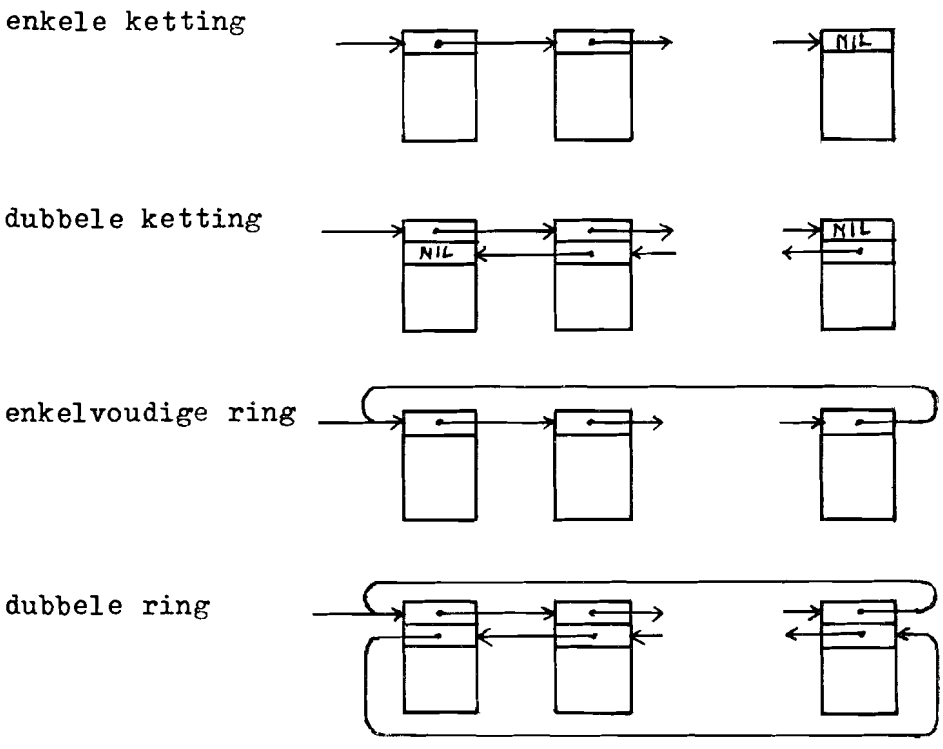
- Moet de pointer gevonden worden, waarmee een bepaald element bereikt kan worden (basis-operatie 4), dan vereist dit het zoeken in het object naar een element, dat aan een bepaalde beschrijving voldoet. Als alle elementen in een object moeten worden afgelopen, zal deze operatie vrij veel tijd kosten. Een object wordt daarom opgesplitst in meerdere collecties, voor elk type element één. Een object kan zodoende bestaan uit maximaal vijf collecties, die

elk een onbeperkt aantal elementen van een type kunnen bevatten.

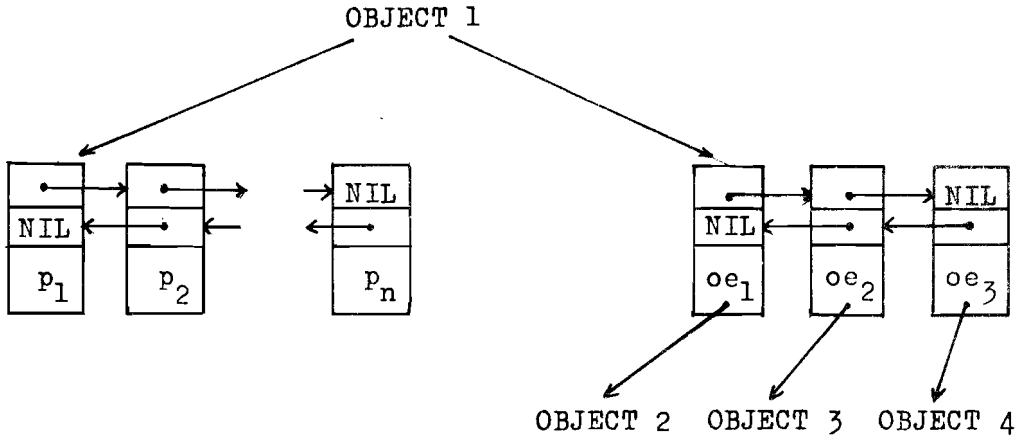


Figuur 5.3. Een object opgebouwd uit collecties van verschillende soorten elementen.

- Er zijn operaties, waarbij een object element voor element moet worden afgelopen. Dit betekent, dat in elke collectie de elementen stuk voor stuk bereikbaar moeten zijn. Voor de structuur in een collectie zijn daardoor ketting-structuren het meest geschikt:



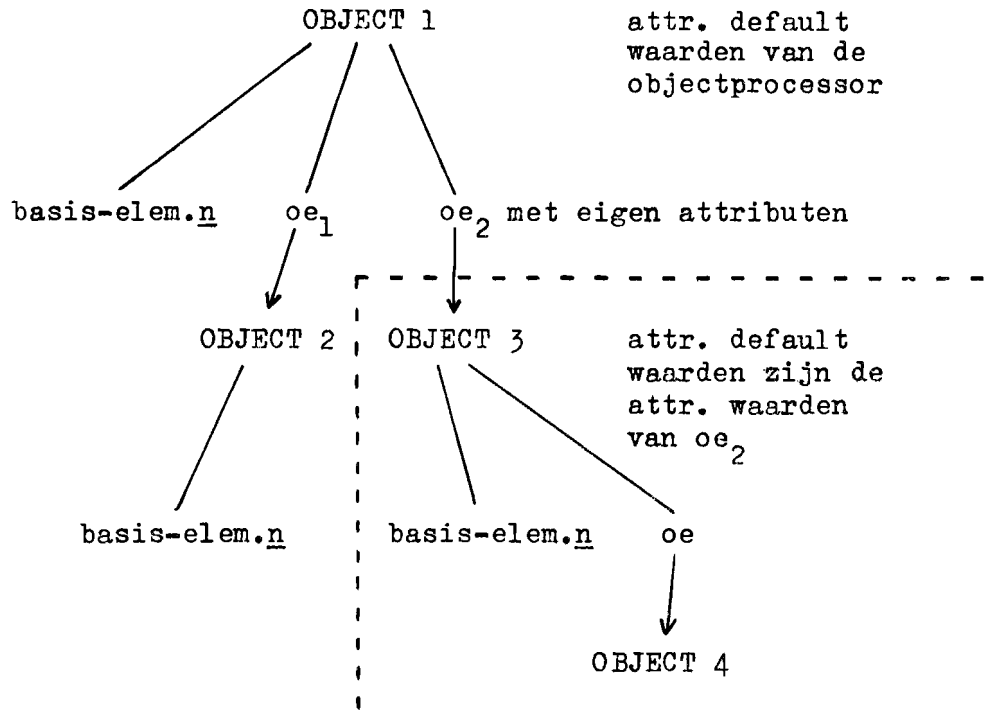
- Bij het kiezen van een van deze ketting-structuren moet rekening worden gehouden met de basis-operaties 2 en 3, d.w.z. er moeten elementen aan kunnen worden toegevoegd en er moeten elementen uit kunnen worden verwijderd. Het toevoegen zal bij de enkele ketting de minste tijd kosten: als een nieuw element aan het begin van de ketting wordt toegevoegd, betekent dit slechts het aanpassen van twee pointers. Het verwijderen van een element geeft met deze structuur moeilijkheden: voor het herstellen van de ketting moet de pointer naar het voorgaande element bekend zijn. Omdat i.v.m. de basis-operaties 1 en 4 noodzakelijk is, dat het laatste element in een collectie aangegeven is, is de dubbele ketting gekozen. Het einde van de ketting kan hierin worden gemarkeerd door de voorwaartse pointer in het laatste element NIL te maken (fig. 5.4).



Figuur 5.4. Voorbeeld van een object-structuur.

5.2.2. De attributen van een element.

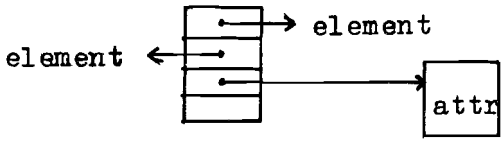
De representatie van elementen wordt bepaald door een aantal attributen. Is een element zonder attributen gedefinieerd, dan wordt de representatie van dat element bepaald door een aantal default-waarden. Deze zijn in de objectprocessor gedefinieerd, of gedefinieerd door de attribuut-waarden van het object-element, dat naar het object verwijst, waartoe het element behoort (zie fig. 5.5)



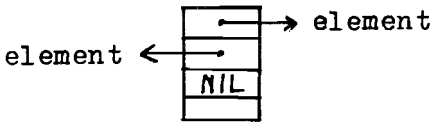
Figuur 5.5. De attribuut default waarden.

Omdat elementen niet altijd met eigen attributen worden gedefinieerd, bestaan voor de attributen aparte knooppunten, die met een element-knooppunt worden verbonden in het geval dat het element een of meerdere eigen attributen heeft:

- element met eigen attributen



- element, dat zonder attributen is gedefinieerd

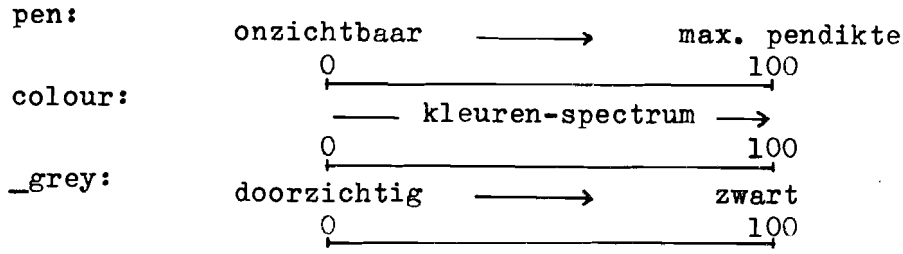


Attributen kunnen relatief of absoluut worden gedefinieerd:
- absolute attributen geven direct de representatie van het element weer
- een relatief attribuut bepaalt samen met de geldende default-

waarde de representatie.

De verschillende attributen kunnen de volgende waarden aannemen:

- pen, colour, cgrey, ogrey
  absolute waarde: 0..100



  relatieve waarde: -100..100

  Bij deze attributen wordt de representatie van het element bepaald door de som van de relatieve waarde en de voor dat element geldende default-waarde (eis: $0 \leq \text{som} \leq 100$)

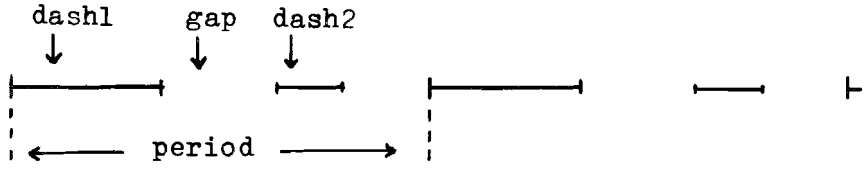
- llst, clst, olst
  M.b.v. dit attribuut kunnen de volgende lijn-typen worden aangegeven:

  - getrokken lijnen
  - gestippelde lijnen
  - gestreepte lijnen
  - streep punt lijnen

  Hiervoor is het data-type 'ldescr' gedefinieerd:

```
ldescr = record
           period : real;
           dashl, gap, dash2 : 0..100
         end; (*ldescr*)
```

  Verklaring van de items

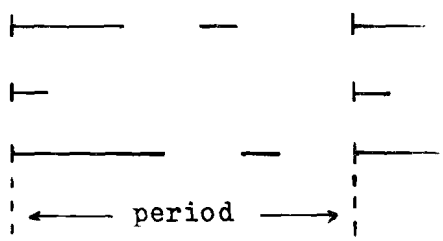


Het item 'period' wordt gedefinieerd in lengte-eenheden van het object-coördinaten-stelsel. De werkelijke lengte van b.v. 'dashl' in dit stelsel is daardoor gedefinieerd als:

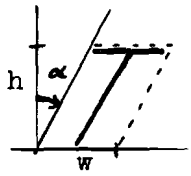
$$\text{dashl} = \frac{\text{period}}{100}.$$

Deze definitie betekent wel, dat bij een scale-operatie op een object de periodes van de verschillende lijnstukken in dat object zullen moeten worden meegetransformeerd.
Een manier, waarop bij dit attribuut uit een relatieve waarde de echte lijn-voorstelling kan worden bepaald, is het optellen van de waarde van het item 'dashl':

default line-style:
relatieve line-style:
werkelijke line-style:



- italic, charh, charw
  Dit zijn uitsluitend attributen voor het text-element.



Voor bovenstaande tekening geldt:

italic = α (graden)
charh = h
charw = w

uitgedrukt in lengte-eenheden van het object-coördinaten-stelsel

Opm.: Bij transformaties zal rekening moeten worden gehouden met de character-afmetingen.

5.2.3. De data-types

Voor de verschillende knooppunten in een object-definitie zijn een aantal data-types gedefinieerd van het type record:

- Object-knooppunt

  Het record, dat hiervoor gedefinieerd is, bestaat uit een aantal pointers naar het eerste element van elke collectie. Bij het ontbreken van een collectie zal de betrokken pointer de waarde NIL bevatten

```
object = record
           plink : ↑element; (*point*)
           llink : ↑element; (*line*)
           clink : ↑element; (*contour*)
           tlink : ↑element; (*text*)
           oelink : ↑element (*object-elem.*)
         end; (*object*)
```

- Element-knooppunt

  Voor de opslag van de informatie van een element is het data-type 'element' gedefinieerd. Dit is een variable record, waarin de varianten bepaald zijn door de verschillende soorten elementen. Alle geometrische informatie over de variabele 'e' kan hierin worden ondergebracht.

```
element = record
            flink : ↑element;
            blink : ↑element;
            ralink : ↑repr;
            case elementtype : elem of
            point : (x, y, z : real);
            line  : (xb, yb, zb, xe, ye, ze : real);
            contour : (coordtrl : ↑coordtr;
                       nofcoordtrs : integer);
            text  : (xlo, ylo, zlo, xro, yro, zro,
                     xlb, ylb, zlb : real;
                     chars : array[1..10] of char);
```

```
            objel : (objref : array[1..10] of char;
                     tmatrix : matrix;
                     index : integer)
          end; (*element*)
```

Toelichting bij het record 'element':

flink : pointer naar volgende element in ketting;
bij het laatste element in de ketting is deze
pointer NIL.

blink : pointer naar voorgaande element in ketting;
bij het eerste element in de ketting is deze
pointer NIL.

ralink : pointer naar attributen-record;
deze pointer is NIL als het element zonder attributen
is gedefinieerd.

index : dit item geeft de index aan van de object-administratie
entry van het door het object-element gerefereerde
object.
Opm.: referenties naar objecten m.b.v. pointers en
namen zijn niet bruikbaar omdat:
- door enkele operaties de locatie van een object
in het geheugen kan veranderen,
- na de operatie REMOBJ een object niet meer
onder een naam bereikbaar mag zijn.
(zie par. 5.3)

- Attributen-knooppunt

Voor opslag van de attributen is het volgende variable record
gedefinieerd:

```
repr = record
        relpen, relcol, rellst, relgrey : boolean;
        detect : boolean;
        pen, colour : -100..100;
        case elementtype : elem of
        line    : (llst : ldescr);
```

```
        contour : (clst : ldescr;
                   cgrey : -100..100);
        text    : (italic, charh, charw : real);
        objel   : (olst : ldescr;
                   ogrey : -100..100)
      end; (*repr*)
```

Of bij het toevoegen van een element aan een object een attributen-record moet worden gecreëerd, kan worden afgeleid uit het item eval in 'e'. Hierin staat vermeld welke items bij de definitie van het element zijn opgegeven. Zijn hier geen attributen bij, dan wordt er geen 'repr'-record gecreëerd. Zijn er wel attributen voor het element gedefinieerd, dan worden deze attributen opgeslagen in de daarvoor bestemde plaatsen in het attributen-record. Voor de ongedefinieerde items van dit record worden default waarden ingevuld. Het record zal bereikbaar zijn via de pointer 'ralink' in het 'element'-record.

Opmerking bij par. 5.2.3:

Voordat definitief voor bovenvermelde definitie is besloten, is onderzocht of een contour in de vorm van driehoeken kan worden opgeslagen. Dit vereiste, dat elke contour voor opslag werd omgezet in driehoeken en omgekeerd, voor weergave, dat uit een aantal driehoeken een contour werd gereconstrueerd.
Als argumentatie voor de keuze van een driehoek is aan te voeren, dat de opslag van oppervlakte-elementen m.b.v. driehoeken niet meer afhankelijk is van het variabele aantal hoekpunten van het contour-element.
Ook zijn er bepaalde grafische operaties (hidden-line elimination), waarbij nuttig gebruik kan worden gemaakt van de driehoekrepresentatie.
Omdat het mogelijk is, dat een object alleen uit driehoeken is opgebouwd, ontstaan er moeilijkheden: er is dan geen onderscheid meer mogelijk tussen deze driehoeken en de driehoeken die in een object een contour voorstellen. Daarom is deze manier van opslag van een contour niet gehandhaafd.

5.2.4. De objectadministratie

De objectprocessor beheert een aantal objecten. Binnen de objectprocessor worden de objecten aangeduid met pointers. Is 'op' de pointer naar een object (pointer type: ↑object) dan wordt het object aangeduid met op↑ (PASCAL). Buiten de objectprocessor zijn de objecten onder een naam bekend. Hierdoor is een administratie nodig, die bij elke naam de pointer naar het desbetreffende object registreert.
Deze objectadministratie is berekend op een totaal van tien objecten en is dan ook gedeclareerd als een array van tien 'objdescr'-records. Een 'objdescr'-record bevat van een object de naam en de locatie (pointer). D.m.v. een integer kan hierin ook nog worden aangegeven, hoeveel object-elementen refereren naar het object.

```
objadm : array[1..10] of objdescr;

objdescr = record
             obj : array[1..10] of char;
             objloc : ↑object;
             nofref : integer
             end : (*objdescr*)
```

De procedures, die in de objectprocessor bestemd zijn om te opereren op de objectadministratie worden in par. 5.3.2. beschreven.

5.3. Implementatie van de operaties

Voor het opereren op de objecten zijn een aantal procedures geschreven. We verdelen de verschillende procedures in de objectprocessor in een aantal niveau's. We maken onderscheid tussen b.v. procedures, die door de besturing van de objectprocessor worden aangeroepen en procedures, die op de object-datastructuur een aantal algemene basis-operaties uitvoeren. Procedures, die enkele belangrijke basis-operaties realiseren, zijn b.v. ADDE, DELE, NEXTEL, RETSPACE en COPYOBJ. Deze procedures zijn beschreven in par. 5.3.1. De procedures, die door het besturingsprogramma kunnen worden aangeroepen, zijn volgens een aantal categoriëen behandeld. Deze categoriëen komen overeen met de functies, die in hoofdstuk 3 zijn opgenoemd.

5.3.1. Procedures t.b.v. basis-operaties

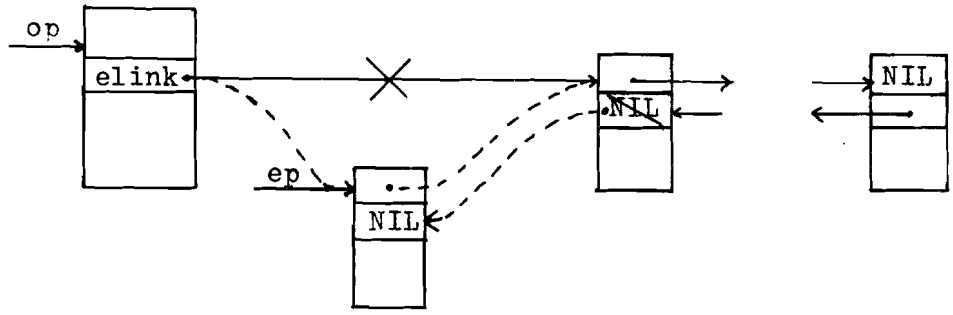
Er bevinden zich een aantal procedures in de objectprocessor, waarmee basis-operaties kunnen worden uitgevoerd (het invoegen van een element in een collectie, het verwijderen van een element uit een collectie en het aflopen van alle elementen in een object). Op het niveau, waarop deze operaties uitgevoerd worden, worden de elementen en objecten aangeduid d.m.v. de pointers 'ep' resp. 'op'.

'ep' is pointer naar element (pointer type: ↑element)
'op' is pointer naar object (pointer type: ↑object)

Opm.: In deze par. wordt met op↑.elink de pointer bedoeld, die vanuit het 'object'-record naar het eerste element van een collectie wijst.

- Procedure ADDE (ep, op);

Als regel is gekozen, dat een element ep↑, dat aan een object op↑ toegevoegd moet worden, vooraan in de ketting wordt geplaatst. (zie fig. 5.6)



Figuur 5.6. Het toevoegen van een element.

Deze operatie vereist de volgende acties:

ep↑.blink:=NIL;

ep↑.flink:=op↑.elink;

op↑.elink↑.blink:=ep;

op↑.elink:=ep.

Een speciaal geval is hierbij, dat op↑.elink=NIL d.w.z. er is nog geen enkel element in die collectie:

ep↑.blink:=NIL;

ep↑.flink:=NIL;

op↑.elink:=ep.

Opm.1:

Het toevoegen van een object-element vereist het ophogen van het item 'nofref' van het gerefereerde object met 1.

Opm.2:

Bij een contour-element worden de coördinaten-triple records m.b.v. de procedure COPYCTRS (newe, ctrp) gecopieerd en (in in de juiste volgorde!) aan het nieuwe contour-element toegevoegd.

- Procedure DELE (ep, op);

Bij het verwijderen van een element ep↑ moet er rekening mee worden gehouden, dat het in de dubbele ketting kan optreden als:

- eerste element,
- laatste element,
- enige element,

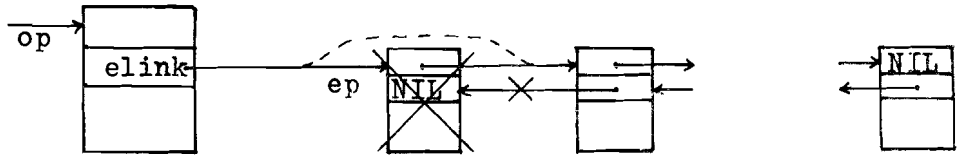
- element, dat zowel een voorganger als een opvolger in de collectie heeft.

De handelingen, noodzakelijk voor het herstellen van de ketting, zijn voor deze gevallen nogal verschillend:

- ep↑ is eerste element:

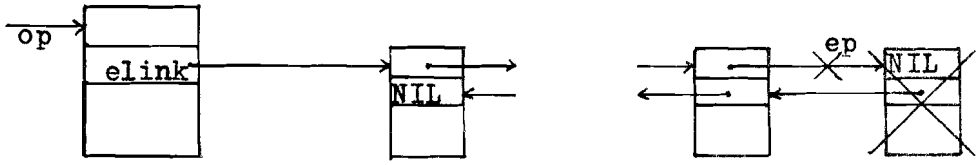
  ep↑.flink↑.blink:=NIL

  op↑.elink:=ep↑.flink.



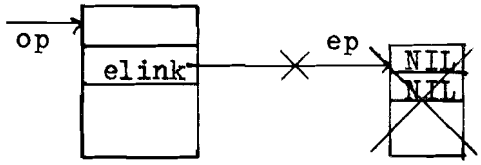
- ep↑ is laatste element:

  ep↑.blink↑.flink:=NIL.



- ep↑ is enige element in collectie:

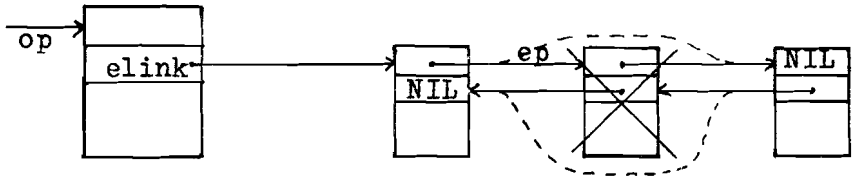
  op↑.elink:=NIL.



- ep↑ is een element, gesitueerd in het midden van de ketting:

  ep↑.blink↑.flink:=ep↑.flink;

  ep↑.flink↑.blink:=ep↑.blink.



Opm.:

Het verwijderen van een object-element vereist het verminderen van het item 'nofref' van het gerefereerde object met 1 en eventueel het verwijderen van het gerefereerde object met RETSPACE (zie par. 5.3.2).

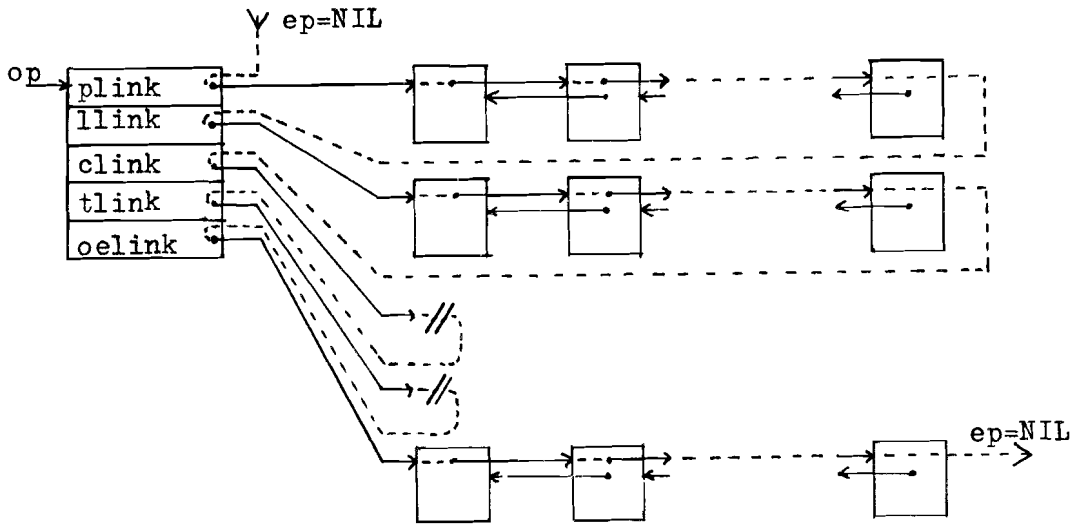
- Procedure NEXTEL (ep, op);

Bij operaties zoals 'TRANSF OBJECT' en 'DISPLAY', moeten de elementen in een object stuk voor stuk geselecteerd worden. Voor de volgorde waarin dat gebeurt, wordt een bepaalde regel ingevoerd. Als uitgangspunt wordt het object-knooppunt gekozen. De verschillende collecties elementen worden afgelopen in de volgorde zoals er in het object-record naar verwezen wordt, dus punten, lijnstukken, contouren, teksten, object-elementen. Het resultaat van een procedure-aanroep NEXTEL (ep, op) is dan:

als ep=NIL : de locatie van het eerste element in object op↑ wordt aan 'ep' toegekend.
Dit betekent, dat 'ep' kan worden: op↑.plink,
op↑.llink,
op↑.clink,
op↑.tlink,
of op↑.oelink.

als ep≠NIL : ep:=ep↑.flink;
is hierdoor ep=NIL (het element was het laatste in zijn collectie), dan zal 'ep' de locatie aan gaan geven van het eerste element in de volgende aanwezige collectie (mits elementtype≠objel).

is ep↑ het laatste element in object op↑, dan ep:=NIL.



Figuur 5.7. De volgorde, waarin de elementen van een object worden afgelopen.

- Procedure RETSPACE (op);

  In deze procedure moet de geheugenruimte worden vrijgegeven van de elementen van object op↑. Dit zou voor elk element ep↑ in object op↑ (NEXTEL) met de PASCAL-statement dispose (ep) uitgevoerd moeten worden. De functie dispose is echter nog niet opgenomen in de huidige PASCAL-compiler, vandaar dat in deze procedure alleen nog maar van belang is, dat voor elk object-element in object op↑ het item 'nofref' van het gerefereerde object met 1 wordt verminderd.

- Procedure COPYOBJ (op, newo);

  De elementen in object op↑ worden m.b.v. NEXTEL afgelopen. Hierbij wordt van elk element een copie gemaakt, dat met de procedure ADDE wordt toegevoegd aan object newo↑.

5.3.2. Het beheren van een aantal objecten

De procedures, die deze functie van de objectprocessor verzorgen, opereren op de objectadministratie.

- INIT;

  Initialisatie betekent voor de variabele 'objadm', dat voor elk 'objadm'-record: obj := 'no object '
  objloc := NIL
  nofref := 0

- DEFOBJ (obj);

  Deze boolean-function wordt aangeroepen als een object wordt gedeclareerd. Hiermee wordt voor het object in 'obj' in de objectadministratie een entry gereserveerd. Het is niet geoorloofd een object te declareren onder een naam, waaronder een ander object in de objectadministratie geregistreerd staat.
  function DEFOBJ: true: registratie heeft plaatsgevonden
  false: geen vrije records in 'objadm'

- ADMLOC (obj, op);

Het object aangewezen door de pointer 'op' wordt in de object-administratie geregistreerd onder de naam, die door 'obj' wordt aangegeven. Dit betekent dat voor de entry, waarvoor geldt objadm[i].obj = obj, objadm[i].objloc := op.
Opm.: Was het item objadm[i].objloc ≠ NIL, dan zal het door deze pointer aangewezen object worden verwijderd met RETSPACE.

- DELOBJ (op);

Zal een object nooit meer gebruikt worden, dan kan het verwijderd worden. Is het item 'nofref' van dat object ≠ 0, dan wordt het object nog in een ander object als object-element gebruikt. Dit betekent dat het object voor dit object-element nog wel bereikbaar moet zijn. Dit gebeurt dan via de index van de objadm-entry van het object.
Dus:

nofref = 0: object wordt gewoon verwijderd, d.w.z.
obj := 'no object '
objloc := NIL

nofref ≠ 0: obj := 'no object '
objloc en nofref worden niet veranderd.
Het object is dus niet meer gekoppeld aan een naam.

- GETIND (obj);

Met deze boolean-function wordt de objadm-entry van het object 'obj' gezocht. De globale variabele 'i' krijgt de index-waarde van deze entry.
function GETIND: true: objectnaam is geregistreerd in 'objadm'
false: objectnaam is niet geregistreerd.

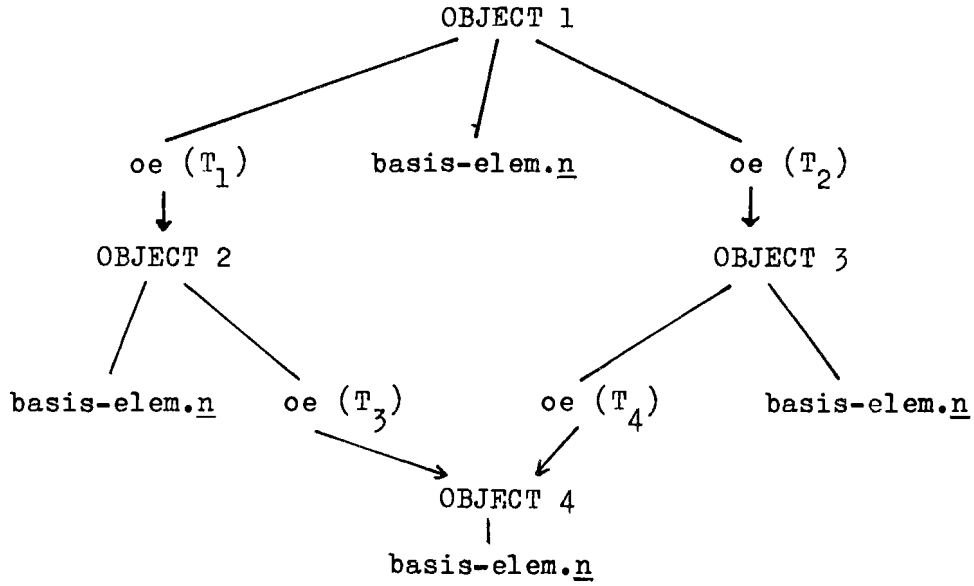
- LOCOBJ (obj, op);

Met GETIND (obj) wordt de index i van de objadm-entry van object obj gezocht. De locatie van het object is dan bekend: objadm[i].objloc. Deze pointer wordt met 'op' afgegeven.

function LOCOBJ: true: objectnaam is geregistreerd in 'objadm'
false: objectnaam is niet geregistreerd.

5.3.3. Het weergeven van een object

De geometrische informatie van alle basis-elementen, die direct of indirect deel uitmaken van het weer te geven object, moet getransformeerd worden naar een twee-dimensionaal coördinaten stelsel. Er wordt voor de viewing transformatie gebruik gemaakt van het GINO-F pakket. Alle coördinaten triples moeten voor deze transformatie gedefinieerd zijn in het coördinaten stelsel van het weer te geven object. Er is daardoor een procedure noodzakelijk, die de gehele object-structuur doorloopt en hierbij van alle basis-elementen de werkelijke drie-dimensionale coördinaten berekent. Voor deze berekening wordt de geometrische informatie van de elementen getransformeerd met een matrix, die het product is van de tmatrices, die vanaf de wortel van de structuur op weg naar het element bij de object-elementen zijn gevonden. Het vermenigvuldigen van de tmatrices gebeurt volgens een bepaalde regel. Aan de hand van de in fig.5.8 weergegeven structuur wordt deze regel gedefinieerd.



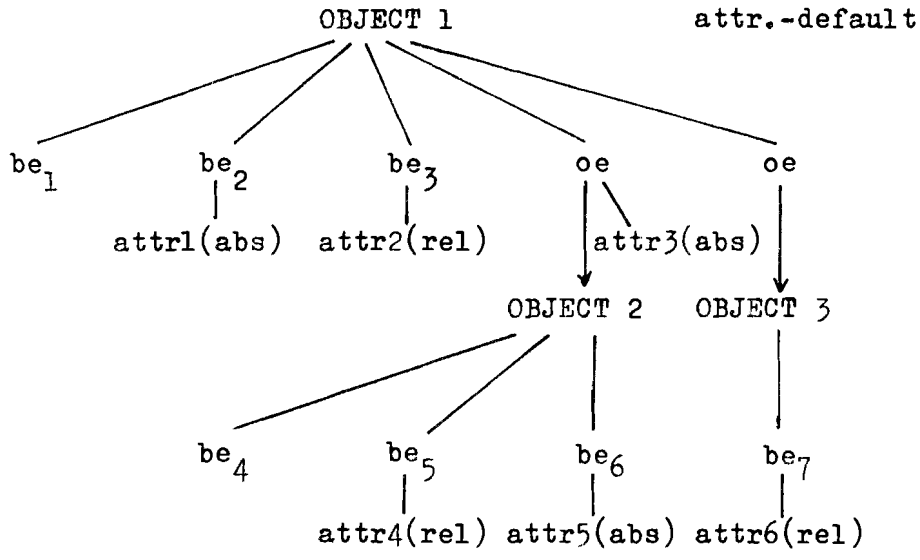
Figuur 5.8. Een object-structuur.

Dit object kan na het transformeren van de door object-elementen gerefereerde objecten als volgt worden weergegeven met uitsluitend basis-elementen:

$$
\begin{aligned}
&(\text{basis-elementen van object1}) \\
+\ &(\text{basis-elementen van object2}) * T_1 \\
+\ &(\text{basis-elementen van object4}) * T_3 * T_1 \\
+\ &(\text{basis-elementen van object3}) * T_2 \\
+\ &(\text{basis-elementen van object4}) * T_4 * T_2
\end{aligned}
$$

Relatieve attributen, die bij een element gedefinieerd zijn, worden eveneens op een dergelijke manier verwerkt. In het in fig. 5.9 weergegeven object is de waarde van het attribuut 'attr' voor elk element:

be_1 : attr-default;
be_2 : attr1;
be_3 : attr-default met rel. waarde attr2;
be_4 : attr3;
be_5 : attr3 met rel. waarde attr4;
be_6 : attr5;
be_7 : attr-default met rel waarde attr6.



Figuur 5.9 . Een objectstructuur met daarin aanwezig een aantal van attributen voorziene elementen.

Door de manier, waarop voor deze operatie de informatie van een object verwerkt moet worden, is een procedure DISPLAY geschreven, die recursief aangeroepen kan worden en die als parameters een matrix, een aantal attribuut-waarden, en een pointer naar een object mee krijgt:

DISPLAY (op, trm, attr. default-waarden);

- De geometrische informatie van elk basis-element in object op↑ wordt getransformeerd met de matrix trm en vervolgens wordt het element met behulp van GINO-F weergegeven.
- Elk object-element in object op↑ veroorzaakt een recursieve aanroep:

  DISPLAY (optr, m, attr. def.)

  optr: pointer naar het gerefereerde object

  m=tmatrix (van object-element) * trm

  attr. def.: attr. default-waarden voor gerefereerde object.

Opm.: Het is mogelijk door herhaald de procedure DISPLAY aan te roepen

1 - verschillende objecten op een scherm weer te geven;
2 - van een object verschillende aanzichten (in viewports) af te beelden.

5.3.4. Het manipuleren met objecten

- ADD (e, op)

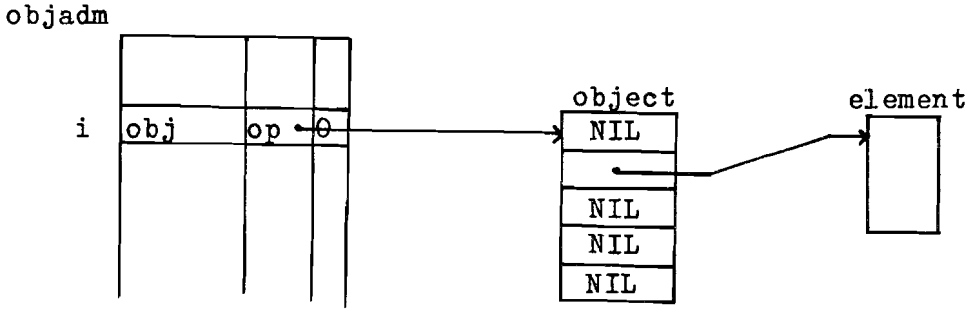
  In deze procedure wordt een element gecreëerd: new(ep).
  Dit element ep↑ wordt gedefinieerd volgens een element-beschrijving in de variabele 'e'. Het wordt toegevoegd aan object op↑ m.b.v. de procedure ADDE (ep, op), als dit object tenminste is gedeclareerd. Is het object gedeclareerd, maar het had nog geen object-record, dan moet dit eveneens worden gecreëerd (dit betekent dat aan het object het eerste element wordt toegevoegd):

  new(op); alle pointers van op↑ worden NIL.

  Na deze operatie is e.lastref de pointer naar het toegevoegde

element en is e.obj de naam van het object op↑.

Opm.: Als een object-element wordt toegevoegd, zal het hierdoor gerefereerde object gedeclareerd moeten zijn. Ook is in dit geval controle op cycles noodzakelijk. Kan het object-element toegevoegd worden dan wordt het item 'nofref' van het gerefereerde object met 1 opgehoogd.



Figuur 5.10. Het toevoegen van het eerste element aan een object.

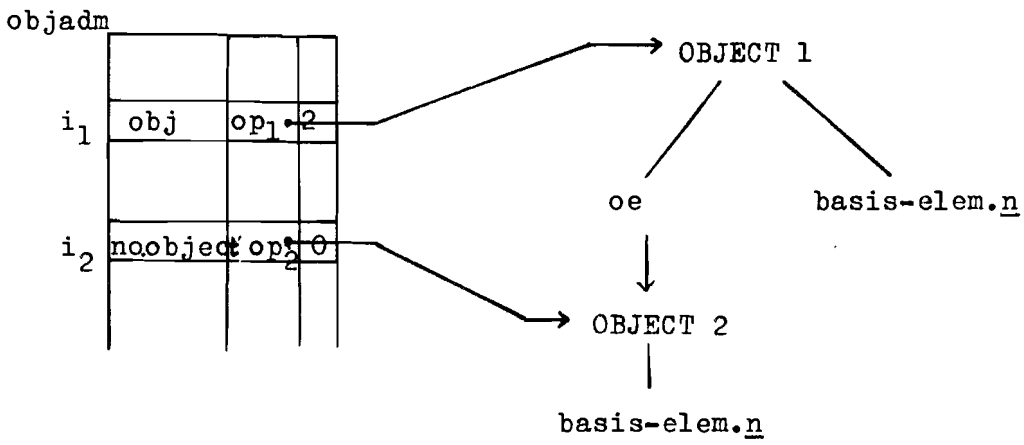
- DEL (e, op);

Voordat een element kan worden verwijderd, zal het eerst in het object gelocaliseerd moeten worden. Er is dus een procedure nodig, waarin in een object op↑ gezocht wordt naar een element, dat overeenkomt met de beschrijving in 'e'. Hiervoor is MEMBER (e, op) geschreven: een boolean-function. Als m.b.v. MEMBER (e, op) een element gevonden is (MEMBER=true), is de pointer naar dat element te vinden in het item 'lastref' van de variabele 'e'. Het gevonden element e.lastref↑ kan dan worden verwijderd met de elementaire operatie DELE (e.lastref, op).

Is er echter sprake van het verwijderen van een object-element, dan moet het item 'nofref' van het gerefereerde object met 1 worden verminderd. De mogelijkheid is nu aanwezig, dat dit object voor de gebruiker al lang geen waarde meer had en dat deze een REMOBJ heeft laten uitvoeren op het object. Er zal dus na elke DEL (object-elem., op) nagegaan moeten worden of de situatie is ontstaan, dat in de objadm-entry van het gerefereerde object:

obj ='no object ' èn nofref = 0.

Is het resultaat van dit onderzoek positief, dan wordt in de object-administratie door het verwijderen van het gerefereerde object m.b.v. RETSPACE (op) het aantal vrije entries met 1 uitgebreid.



Figuur 5.11. De procedure aanroep DEL (object.elem, op_1) geeft in deze situatie aanleiding tot RETSPACE (op_2) en het vrijgeven van de entry van $op_2\uparrow$.

- MEMBER (e, op): boolean;

  Het vergelijken van alle items van ieder element met de overeenkomstige items van de variabele 'e', zal ook al zijn de zoekacties naar een bepaald element beperkt tot een collectie, veel tijd kosten.
  Er is echter een manier om de zoektijd naar een element te reduceren. Als we er van uitgaan, dat elk element zijn karakteristieke informatie heeft, dan betekent dit, dat slechts daarop gecontroleerd hoeft te worden. De variabele 'e' beschikt nu voor elk type element over een set, waarmee de items, die op deze informatie betrekking hebben, kunnen worden aangegeven (zie par. 5.1). Er hoeft dan slechts op die items gecontroleerd te worden. Is een element gevonden, dat aan het in 'e.eval' gedefinieerde gelijkheidscriterium voldoet, dan wordt de locatie ervan in het item 'lastref' van 'e' geplaatst en krijgt de functie MEMBER de waarde true.

- EMPTY (op);

  Het effect van deze procedure aanroep:
  1 - De geheugenruimte, die door de elementen van object op↑ wordt ingenomen, wordt m.b.v. RETSPACE (op) vrijgegeven.
  2 - Alle pointers in het object-record worden NIL.

- UNION (op_1, op_2);
  Het object, waarvan de naam in 'obj' staat, bestaat na UNION uit alle elementen van object $op_1\uparrow$ èn alle elementen van object $op_2\uparrow$ (zie par. 3.2). We onderscheiden de volgende stappen:
  1 - Het creëren van een object-record (new(newo)).
  2 - M.b.v. COPYOBJ (op_1, newo) alle elementen van object $op_1\uparrow$ aan newo$\uparrow$ toevoegen.
  3 - M.b.v. COPYOBJ (op_2, newo) alle elementen van object $op_2\uparrow$ aan newo$\uparrow$ toevoegen.
  4 - De nieuwe object-definitie bereikbaar maken door met ADMLOC (obj, newo) de locatie newo te plaatsen in het objadm-record van object obj.

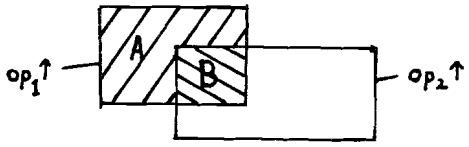
- DIFF (op_1, op_2);
  Het object, waarvan de naam in 'obj' staat, bestaat na DIFF uit die elementen van $op_1\uparrow$, die niet in object $op_2\uparrow$ aanwezig zijn (zie par 3.2).
  Hiervoor zal achtereenvolgens worden uitgevoerd:
  1 - Het creëren van een object-record (new(newo)).
  2 - M.b.v. COPYOBJ (op_1, newo) alle elementen van object $op_1\uparrow$ hieraan toevoegen.
  3 - Voor elk element in object $op_2\uparrow$ nagaan of in newo$\uparrow$ een identiek element aanwezig is en zo ja dit element verwijderen. (functie MEMBERE: zie hierna).
  4 - De nieuwe object-definitie voor de gebruiker bereikbaar maken door met ADMLOC (obj, newo) de locatie newo te plaatsen in het objadm-record van object obj.
  Bij de derde stap kan geen gebruik worden gemaakt van de procedure MEMBER (e, newo), omdat de elementinformatie steeds aanwezig is in een record van het type 'element'. Daarom is een functie MEMBERE (ep, idep, newo) geschreven, waarin in object newo$\uparrow$ gezocht wordt naar een element, dat voor zowel geometrische informatie als attributen identiek is aan het element ep$\uparrow$. Is zo'n element gevonden (MEMBERE = true), dan wordt het verwijderd met DELE (idep, newo).

- INTERSECT (op_1, op_2);

  Deze operatie is te realiseren d.m.v. twee opeenvolgende verschil-operaties.

  DIFF (op_1, op_2); (* resultaat: object op↑ *)
  DIFF (op_1, op).

  Met onderstaande tekening wordt dit nu duidelijk gemaakt voor objecten zonder identieke elementen.



  Na de eerste verschil-operatie wordt aan het object 'obj' een objectdefinitie toegekend, die overeenkomt met de elementen in gebied A van bovenstaande tekening. Na de tweede verschil-operatie bevat dat object een objectdefinitie, die overeenkomt met de elementen in gebied B.

- TRAFO (op_1, tm);

  Om een nieuw object op te bouwen, dat kan worden beschouwd als de getransformeerde versie van object op_1↑, moet van elk element in object op_1↑ een getransformeerde copie aan een nieuw object worden toegevoegd. Dit leidt tot de volgende operaties:

  1 - Open een nieuw object door het creëren van een object-record (new(newo)).
  2 - Van elk element in op_1↑ wordt een copie gemaakt, waarvan de geometrische informatie (coördinaten triples / tmatrices) getransformeerd wordt m.b.v. de matrix tm; de getransformeerde copie wordt toegevoegd aan het object newo↑.
  3 - Het nieuwe object wordt d.m.v. ADMLOC (obj, newo) geregistreerd in de object-administratie onder de naam 'obj'.

  Het is belangrijk, dat de vermenigvuldiging met de matrix tm volgens een bepaalde regel geboeurt. Als we naar de procedure DISPLAY teruggaan, zien we dat daar al een afspraak is gemaakt. Door deze hier over te nemen, wordt het resultaat van een

transformatie:
bij een coörd.triple (xt, yt, zt, t):=(x, y, z, 1) * tm
bij een tmatrix tmatrix':= tmatrix * tm.

- WRITEOBJ (op);
Dit is een procedure, waarmee het object op↑ wordt uitgeschreven, zoals het in de data-structuur aanwezig is. Als object-elementen in op↑ voorkomen, wordt de procedure recursief aangeroepen met als argument de locatie van het gerefereerde object. Er worden zodoende meerdere object-beschrijvingen afgegeven, die elk vergezeld gaan van de tmatrix, die de transformatie op dat object beschrijft. De procedure kan vergeleken worden met DISPLAY, echter met dit verschil dat hier geen transformatie plaatsvindt.
Een voorbeeld van een beschrijving is gegeven onder deze operatie in H6.

5.3.5. Opslag van een object.

Een object moet voor langere tijd op een achtergrondgeheugen kunnen worden opgeslagen. Operaties hiervoor zijn STOREOBJECT en GETOBJECT. Hoewel in mijn programma deze operaties niet zijn opgenomen, zal ik hieronder toch kort beschrijven hoe eventueel bij deze operaties te werk kan worden gegaan.

STOREOBJECT (op, ofile);
Het object op↑ wordt element voor element afgelopen en de informatie, die hierbij in elk record gevonden wordt, zal in een file worden ondergebracht. Komen er object-elementen in het object voor, dan betekent dit dat in de file ook de beschrijvingen aanwezig moeten zijn van die objecten, waarnaar verwezen wordt.

GETOBJECT (ofile);
Doordat na het terugvragen van de in de file aanwezige informatie de objecten weer onder hun oorspronkelijke naam geregistreerd moeten zijn, zal in de file elke objectbeschrijving vergezeld

moeten gaan van zijn oorspronkelijke naam. Voordat een object-element bij deze operatie opnieuw gecreëerd kan worden, moet het gerefereerde object in verband met de objadm-index al in de administratie zijn opgenomen. Door nu aan het begin van de file alle namen te vermelden van de objecten, die erin voorkomen, kunnen de gerefereerde objecten worden gedeclareerd, voordat aan de eigenlijke reconstructie van het object begonnen wordt.

6. DE BESTURING VAN DE OBJECTPROCESSOR

Er zijn verschillende manieren waarop de objectprocessor bestuurd kan worden. Om de objectprocessor uit te testen, is gekozen voor een vraag-antwoord besturing via een teletype : is een invoergegeven door de objectprocessor verwerkt, dan zal kenbaar worden gemaakt wat als volgende invoer wordt verwacht.
Invoergegevens kunnen zijn: objectnamen, elementbeschrijvingen, transformatiegegevens en operatiecommando's. Bij de aanvang van een zitting is de objectadministratie al automatisch geinitialiseerd. Om operatie-commando's wordt gedurende de zitting gevraagd met de tekst 'next command'.
Afhankelijk van de operatie moeten bepaalde gegevens worden ingevoerd. Deze invoer wordt geregeld door speciale procedures:

- een procedure voor het inlezen van een objectnaam.
- procedures voor het invoeren van element-beschrijvingen.
- een procedure voor het opbouwen van een transformatie matrix.

In dit hoofdstuk wordt met een streep onder characters van een commando aangegeven, dat alleen die characters noodzakelijk zijn voor de juiste interpretatie van het commando.(de eerste drie characters). Bij de beschrijving van de dialogen, wordt tekst van de objectprocessor-besturing voorafgegaan door het teken '?'.

6.1. Procedures voor informatie-invoer

6.1.1. Het inlezen van een objectnaam (READOBJ (op))

Aan de gebruiker wordt met 'enter objectname' verzocht een objectnaam in te voeren. Deze objectnaam wordt ondergebracht in de variabele 'obj', waarna een geslaagde invoer-operatie met 'objectname: (obj)' wordt bevestigd. Het current object is dan het ingevoerde object (obj, op, t). Gecontroleerd wordt nog of het object in de object-administratie geregistreerd is en het resultaat hiervan zal afhankelijk van de operatie verder verwerkt worden: bij NEW-OBJECT mag er geen objadm-entry met die naam bestaan, terwijl bij de rest van de operaties de naam juist wel gedeclareerd moet zijn.

6.1.2. De invoer van element-gegevens.

Omdat voor de elementen nogal verschillende informatie moet worden ingevoerd, is voor elk type element een afzonderlijke procedure geschreven. Bij de invoer van element-items moet bij elke procedure rekening worden gehouden met de operatie, waarvoor de procedure is ingeschakeld. Bij de operatie ADDELEMENT dient een complete definitie te worden ingevoerd, terwijl bij de operaties DELELEMENT en MEMBER toegestaan is, dat slechts enkele (karakteristieke) gegevens worden gedefinieerd. Er is in de variabele 'e' een speciaal item (eval) aanwezig, waarmee de gedefinieerde items kunnen worden aangegeven. Wordt nu een van de procedures aangeroepen in verband met de operatie ADDELEMENT, dan moet gecontroleerd worden of een volledige element-definitie wordt opgegeven. Bij elke procedure-aanroep i.v.m. deze operatie wordt de noodzaak van deze controle aangegeven met de waarde true voor de parameter 'adde (bij andere operaties false).
Voordat wordt verdergegaan met de afzonderlijke beschrijving van de procedures, kan vanwege het algemene karakter nog dit worden opgemerkt: elke procedure begint met de vermelding 'begin of (type)-description' en eindigt met 'end of (type)-description'.

READPOINT (e, adde);

Om de invoer van een element-item wordt steeds gevraagd met de tekst 'enter point-item'. Mogelijke reacties hierop :

```
x = .. ;
y = .. ;
z = .. ;
pen = .. ;
colour = .. ;
end (* end of point-description*)
```

READLINE (e, adde);

Om de invoer van een line-item wordt steeds gevraagd met de tekst 'enter line-item'. Mogelijke reacties hierop:

```
xb = .. ;
yb = .. ;
zb = .. ;
xe = .. ;
ye = .. ;
ze = .. ;
pen = .. ;
colour = .. ;
lstyle
    ?enter period
      ..
    ?enter dash1
      ..
    ?enter gap
      ..
    ?enter dash2
      ..
end (* end of line-description *)
```

READCONTOUR (e, adde);

Om de invoer van een contour-item wordt steeds gevraagd met de tekst 'enter contour-item'. Mogelijke reacties hierop:

coordtriples
?how many coordinate-triples? (N>3)
N
?enter x-value for point no. 1
..
?enter y-value for point no. 1
..

?enter z-value for point no. N
..
?input of points completed

pen = .. ;
colour = .. ;
lstyle
?enter period
..
?enter dash1
..
?enter gap
..
?enter dash2
..

greyscale = .. ;
end (* end of contour-description *)

READTEXT (e, adde);

Om de invoer van een text-item wordt steeds gevraagd met de tekst 'enter text-item'. Mogelijke reacties hierop:

position
?enter xlo-value
..
?enter ylo-value
..

```
   ?enter zlo-value
       ..
   ?further definition of text-area - yes or no?
no (* bij het antwoord yes: definitie van schrijfrichting en
                                    schrijfvlak *)
string
   ?enter textstring
       ..
pen = .. ;
colour = .. ;
cwidth = .. ;
cheight = .. ;
italicity = .. ;
end (* end of text-description *)
```

READOBJEL (e, adde);

Om de invoer van een object-element item wordt steeds gevraagd met de tekst 'enter object-element item'. Mogelijke reacties hierop:

```
object
   ?enter name of ref. object
       ..
tmatrix
   (* definitie van transformatie-matrix - par.6.1.3 *)
pen = .. ;
colour = .. ;
lstyle = .. ;
   ?enter period
       ..
   ?enter dash1
       ..
   ?enter gap
       ..
   ?enter dash2
       ..
greyscale = .. ;
end (* end of object-el. description *)
```

6.1.3. Tmatrix-definitie (CREATEMATRIX (tm));

Dit is een procedure, waarmee een transformatie-matrix wordt opgebouwd door het invoeren van gegevens over rotatie-, translatie- en scale-operaties in een volgorde, zoals ze op een object moeten worden uitgevoerd. Hiervoor is in de procedure steeds een 'actuele' matrix 'm' aanwezig, die in het begin als een I-matrix gedefinieerd is en dan verder na elke ingevoerde transformatie t wordt gecorrigeerd (m:=m*t). Deze matrix vertegenwoordigt op een gegeven ogenblik alle operaties, die vanaf de aanroep van de procedure zijn gedefinieerd. De procedure begint met de vraag of er wel een transformatie moet worden gedefinieerd: 'transformation-definition? - yes or no'. Als met 'no' is geantwoord, is de door de procedure afgegeven matrix 'tm' slechts een I-matrix. Wordt een transformatie gewenst, dan wordt het begin van de definitie hiervan door het programma aangekondigd met 'begin of tmatrix-definition'.
Voor het opstellen van een matrix, die een bepaalde aaneenschakeling van rotaties, translaties en scale-operaties gaat vertegenwoordigen, zal het programma steeds naar de volgende transformatie hiervoor vragen. Dit gebeurt met 'enter : rotation, translation, scale or endm'. De wijze, waarop na de ontvangst van de verschillende antwoorden door het programma wordt gereageerd is als volgt:

na rotation:
   ?which axis?
   x (, y or z)
   ?how many degrees?
     ..
   ?enter : rotation, translation, scale or endm

na scale:
   ?enter sx-value
     ..
   ?enter sy-value
     ..
   ?enter sz-value
     ..
   (* als t.o.v. een bepaalde coörd.-as geen scale-operatie gewenst

```
      is, voer dan voor de corresponderende s-waarde 1 in *)
   ?enter : rotation, translation, scale or endm
```

na translation

```
   ?enter tx-value
     ..
   ?enter ty-value
     ..
   ?enter tz-value
     ..
   (* als t.o.v. een bepaalde coörd.-as geen translatie gewenst
      is, voer dan voor de corresponderende t-waarde 0 in *)
   ?enter : rotation, translation, scale or endm
```

na endm

```
   ?end of tmatrix-definition
```

Na de ontvangst van endm, wordt de beëindiging van de procedure bekend gemaakt met de tekst 'end of tmatrix-definition'.
Er is dan een matrix verkregen, die bij vermenigvuldiging van de geometrische informatie van een object dezelfde resultaten afgeeft als alle afzonderlijke operaties, die gedurende de procedure zijn gedefinieerd.

6.2. Besturings-commando's

INITIALIZE :

Ofschoon het initialiseren van de object-administratie al automatisch in het begin plaatsvindt, wordt de gebruiker toch in staat gesteld de hiervoor geschreven procedure INIT uit te laten voeren om dan vervolgens opnieuw met het opbouwen van objecten te kunnen aanvangen.

NEWOBJECT :

Invoer :

objectnaam

operatie :

In de object-administratie wordt gecontroleerd of er al een object met die naam geregistreerd is. Is dit niet het geval, dan wordt het object een vrije entry toegewezen. Het object is dan gedeclareerd.

dialoog :

```
?NEXT COMMAND
NEWOBJECT
?ENTER OBJECTNAME
NAAM
?OBJECTNAME :  NAAM
?NEXT COMMAND
```

REMOBJECT :

Invoer :

objectnaam

operatie :

Het object is na deze operatie niet meer met een naam te refereren. De object-definitie kan nog wel voorkomen in andere objecten als object-element.

dialoog :

?NEXT COMMAND

REMOBJECT

?ENTER OBJECTNAME

NAAM

?OBJECTNAME : NAAM

?OBJECT NAAM REMOVED

?NEXT COMMAND

ADDELEMENT :

Invoer :

objectnaam

evt. element-beschrijving

operatie :

De variabele 'e' moet voor deze operatie over de complete definitie van een element beschikken (attr. niet noodzakelijk). Er is voor elk type element een procedure geschreven, die de invoer van element-gegevens regelt en die deze gegevens plaatst in de variabele 'e'. De keuze van de juiste procedure wordt bepaald door het elementtype, zodat dit nog eerst opgegeven moet worden. Gekozen kan worden uit point, line, contour, text, object-elem. en e; e geeft aan, dat het current element moet worden toegevoegd (geen invoer van een of andere element-beschrijving; het element beschreven in 'e' wordt toegevoegd) Na deze operatie is het toegevoegde element het current element geworden.

dialoog :

?NEXT COMMAND

ADDELEMENT

?ENTER OBJECTNAME

NAAM

?OBJECTNAME : NAAM

?ENTER TYPE OF ELEMENT

POINT (of LINE, CONTOUR, TEXT, OBJECT-ELEM, E)

(* bij p,l,c,t,o : invoer van element-gegevens *)

?NEXT COMMAND

DELELEMENT :

Invoer :

objectnaam

evt. elementgegevens

operatie :

Er zijn twee mogelijkheden:

- er wordt een element verwijderd, dat voldoet aan een beschrijving, die aan het programma is opgegeven m.b.v. een van de procedures readpoint, readline, readcontour, readtext, readobjel
- of het current element wordt verwijderd (aanwezig in de variabele 'e').

In het eerste geval moet in het object gezocht worden naar een element, dat beantwoordt aan de ingevoerde gegevens. De operatie zal het snelst kunnen worden uitgevoerd, als zo min mogelijk items zijn gedefinieerd (alleen de karakteristieke informatie van het element). Als zo'n element is gevonden, wordt het verwijderd. Dit wordt dan meegedeeld met de tekst 'equal element found'.

dialoog :

?NEXT COMMAND

DELELEMENT

?ENTER OBJECTNAME

NAAM

?OBJECTNAME : NAAM

?ENTER TYPE OF ELEMENT

POINT (of LINE, CONTOUR, TEXT, OBJECT-ELEM, E)

(* bij p,l,c,t,o : invoer van element-gegevens *)

?EQUAL ELEMENT FOUND

?NEXT COMMAND

MEMBER :

Invoer :

objectnaam

evt. elementgegevens

operatie :

De invoer van (karakteristieke) gegevens van een element wordt door een van de in par. 6.1.2 beschreven procedures geregeld. Evenals bij DELELEMENT kan ook hier zonder de invoer van element-gegevens volstaan worden, waarbij de operatie wordt uitgevoerd aan de hand van de in de variabele 'e' aanwezige informatie. Na de beëindiging van de operatie wordt met

'equal element found'

'element in object objname'

bekend gemaakt, dat het element in het object gevonden is. Het gevonden element is dan het current element geworden.

dialoog :

```
?NEXT COMMAND
MEMBER
?ENTER OBJECTNAME
NAAM
?OBJECTNAME : NAAM
?ENTER TYPE OF ELEMENT
POINT (of LINE, CONTOUR, TEXT, OBJECT-ELEM, E)
(* bij p,l,c,t,o, : invoer van element-gegevens *)
?EQUAL ELEMENT FOUND
?ELEMENT IN OBJECT NAAM
?NEXT COMMAND
```

EMPTYOBJECT :

Invoer :

objectnaam

operatie :

object wordt leeg

dialoog :

```
?NEXT COMMAND
EMPTYOBJECT
?ENTER OBJECTNAME
NAAM
?OBJECTNAME : NAAM
?OBJECT NAAM        IS EMPTY
?NEXT COMMAND
```

UNION :

Invoer :

drie objectnamen (identieke namen toegestaan)

operatie :

Als we ons de operatie voorstellen als UNION (OBJ_1, OBJ_2, OBJ_3) (zie par 3.2), dan moet de volgorde van invoer zijn: OBJ1, OBJ2, OBJ3 of OBJ2, OBJ1, OBJ3. OBJ3 is dan het current object, dat het resultaat van deze operatie krijgt toegewezen.

dialoog :

?NEXT COMMAND
UNION
?ENTER OBJECTNAME (* source 1 *)
OBJ1
?OBJECTNAME : OBJ1
?ENTER OBJECTNAME (* source 2 *)
OBJ2
?OBJECTNAME : OBJ2
?ENTER OBJECTNAME (* destination *)
OBJ3
?OBJECTNAME : OBJ3
?NEXT COMMAND

DIFFERENCE :

Invoer :

drie objectnamen (identieke namen toegestaan)

operatie :

De verschil-operatie DIFF (OBJ1, OBJ2, OBJ3) heeft als effect, dat het object OBJ3 wordt gedefinieerd uit de elementen van object OBJ1, die niet van object OBJ2 deel uit maken.
De volgorde, waarin de namen moeten worden ingevoerd, is OBJ1, OBJ2, OBJ3. OBJ3 wordt daardoor het current object (zie par 3.2).

dialoog :

?NEXT COMMAND
DIFFERENCE
?ENTER OBJECTNAME (* source 1 *)
OBJ1
?OBJECTNAME : OBJ1

?ENTER OBJECTNAME (* source 2 *)
OBJ2
?OBJECTNAME : OBJ2
?ENTER OBJECTNAME (* destination *)
OBJ3
?OBJECTNAME : OBJ3
?NEXT COMMAND

INTERSECTION :

Invoer :
drie objectnamen (identieke namen toegestaan)
operatie :
Als de drie objectnamen in de volgorde OBJ1, OBJ2, OBJ3 worden opgegeven, zal OBJ3 als current object het resultaat van deze operatie toegewezen krijgen (zie par. 3.2).
dialoog :
?NEXT COMMAND
INTERSECTION
?ENTER OBJECTNAME (* source 1 *)
OBJ1
?OBJECTNAME : OBJ1
?ENTER OBJECTNAME (* source 2 *)
OBJ2
?OBJECTNAME : OBJ2
?ENTER OBJECTNAME (* destination *)
OBJ3
?OBJECTNAME : OBJ3
?NEXT COMMAND

TRANSFOBJECT :

Invoer :
twee objectnamen (identieke namen toegestaan)
transformatie-gegevens
operatie :
Het eerste object wordt getransformeerd en aan het tweede zal de getransformeerde informatie worden toegekend. Bij de transformatie is van belang, dat op een eenvoudige manier de trans-

formatie-matrix kan worden gedefinieerd. Hiervoor wordt de procedure CREATEMATRIX gebruikt, die de vereiste informatie voor het opbouwen van de matrix opvraagt. Bij het samenstellen van de matrix in deze procedure wordt uitgegaan van een I-matrix, zodat zonder invoer van transformatie-gegevens het nieuwe object gewoon een copie is van het oude. De volgorde, waarin de rotaties, translaties en scale-operaties worden ingevoerd, is bepalend voor de volgorde, waarin de operaties op het object worden uitgevoerd.

dialoog :

```
?NEXT COMMAND
TRANSFOBJECT
?ENTER OBJECTNAME (* source *)
OBJ1
?OBJECTNAME : OBJ1
?ENTER OBJECTNAME (* destination *)
OBJ2
?OBJECTNAME : OBJ2
(* definitie van transformatie-matrix - zie par. 6.1.3 *)
?NEXT COMMAND
```

WRITEOBJECT :

Invoer :

objectnaam

operatie :

De informatie van zowel de basis-elementen van het opgegeven object als de basis-elementen van de door de object-elementen gerefereerde objecten wordt uitgeschreven. Deze laatste basis-elementen worden dan niet beschreven in de coördinaten zoals ze in werkelijkheid aanwezig zijn in het opgegeven object, maar met hun oorspronkelijke coördinaten en de tmatrix, die de transformatie erop beschrijft (tmatrix van object-element)

dialoog :

```
?NEXT COMMAND
WRITEOBJECT
?ENTER OBJECTNAME
NAAM
?OBJECTNAME : NAAM
```

```
?DESCRIPTION OF OBJECT : NAAM
POINT
  X= 1; Y= 1; Z= 2
  NO ATTRIBUTES
OBJECT-ELEMENT
REFERENCE TO OBJECT : NAAM 2
TRANSFORMATION-MATRIX =
  0  -1   0   0
  1   0   0   0
  0   0   1   0
  5  10   0   1
   NO ATTRIBUTES
?DESCRIPTION OF OBJECT : NAAM 2
LINE
 XB= -1; YB=  0; ZB=  1
 XE= -5; YE=  0; ZE=  1
  NO ATTRIBUTES
?END OF NAAM 2
?END OF NAAM
?NEXT COMMAND
```

DISPLAY (zie par 2.1):

Invoer :

voor elk object, dat moet worden weergegeven:

- objectnaam
- viewing point V
- punt D (viewing direction)
- perspectief/axonometrisch

operatie :

Een aanzicht van het object, gezien vanuit het viewing point V als in de richting van D gekeken wordt, wordt op een plotter getekend. Het is mogelijk dat verschillende aanzichten van het object en ook meerdere objecten op een tekening worden weergegeven. Deze operatie is nog niet uitgetest.

dialoog :

```
?NEXT COMMAND
DISPLAY
?ENTER OBJECTNAME
```

```
    NAAM
    ?OBJECTNAME : NAAM
    ?ENTER VX-VALUE
     ..
    ?ENTER VY-VALUE
     ..
    ?ENTER VZ-VALUE
     ..
    ?ENTER DX-VALUE
     ..
    ?ENTER DY-VALUE
     ..
    ?ENTER DZ-VALUE
     ..
    ?PERSP/AXON?
    AXON
    ?MORE OBJECTS TO DISPLAY? - YES/NO
    NO
    ?NEXT COMMAND
```

END

Dit commando beëindigt de zitting.

dialoog :

```
    ?NEXT COMMAND
    END
    ?END OF SESSION
```

7. NABESCHOUWING

Bij het ontwerpen van de objectprocessor is geprobeerd deze zo gestructureerd mogelijk op te bouwen. Dat het geheel wat betreft rekentijd zo optimaal mogelijk zou moeten functioneren, is daarbij niet het uitgangspunt geweest.
Het uiteindelijke resultaat bevat het grootste deel van de operaties, die in hoofdstuk 3 genoemd zijn. Wat nog niet geschreven is, zijn slechts operaties, die voor het vervullen van de belangrijkste functies niet noodzakelijk zijn. Zo is nog geen procedure geschreven, waarmee de box-operatie kan worden gerealiseerd en is de opslag van objecten in files eveneens nog niet mogelijk.
De objectprocessor is uitgetest m.b.v. een interactief programma, waarbij de communicatie met de gebruiker plaatsvindt d.m.v. een teletype. Objecten, die door de objectprocessor worden beheerd, kunnen d.m.v. subroutines van het pakket GINO-F op een grafisch apparaat zichtbaar worden gemaakt. Interactie op een manier, waarbij b.v. op een scherm meteen de verandering te zien is, die door een operatie in het afgebeelde object is aangebracht, is voor een grafisch systeem wel gewenst. Het is in het hier gebruikte besturingsprogramma echter nog niet opgenomen. Is een verandering in een object aangebracht, dan zal het resultaat pas zichtbaar kunnen worden als het betrokken object opnieuw is getekend (operatie DISPLAY). Wordt de objectprocessor door een bepaald applicatie-programma gebruikt, dan is het altijd mogelijk deze vorm van interactie in te voeren (na een verandering b.v. automatisch: ERASE - DISPLAY).
Er is al gezegd, dat bij het schrijven van de objectprocessor niet is gezocht naar de meest optimale oplossing wat betreft de rekentijd. Tot de veranderingen, waardoor misschien bepaalde operaties sneller kunnen worden uitgevoerd, kan b.v. het onderbrengen van de attributen in het element-record worden gerekend. Dit zal in sommige gevallen (elk element bezit eigen attributen) gunstige resultaten ten gevolge hebben. Wordt echter elk element zonder attributen gedefinieerd (default-value), dan is het in de objectprocessor ingevoerde principe van een apart attributen-record veel

efficiënter. Om een objectprocessor te krijgen, die voor alle toepassingen zo optimaal mogelijk functioneert, zal een grondig onderzoek nodig zijn. Met de objectprocessor zoals die nu is geschreven, is in ieder geval het manipuleren met objecten mogelijk. De volgende stap moet dan een onderzoek zijn naar de bruikbaarheid van de objectprocessor voor een grafisch systeem.

8. LITERATUUR-OVERZICHT

8.1. Literatuurlijst

[1] Sproull/Newman — BP7347bsr
Principles of interactive computer graphics

[2] Encarnacao, J.L. — BP75100bsr
Computer graphics

[3] Nake/Rosenfeld — BP72103bsr
Graphic languages

[4] Walker/Gurd/Drawneek — BP75245bsr
Interactive computer graphics

[5] Prince — EO7122bse
Interactive graphics for CAD

[6] ACM Siggraph
Sigplan Vol. 11 nr. 6
Computer graphics Vol. 10 nr. 1 editor T. Berk
Computer graphics Vol. 10 nr. 2 editor U.W. Pooch
Computer graphics Vol. 9 nr. 1 editor A.P. Lucido

[7] Enkele tijdschriften:
Computer graphics ACM
Computer aug. 1976
Computer and graphics
C.A.D.
On line 1972 blz. 383 e.v. — BP7296bsr

8.2. Inhouds-beschrijving van de literatuur

Van de in par. 8.1 vermelde literatuur geeft Sproull/Newman [1] een goed algemeen overzicht van computer graphics. Het boek behandelt achtereenvolgens:

- display devices
- display files
- interactive graphics
- three-dimensional computer graphics
- graphic systems

Het eerste hoofdstuk geeft een beschrijving van de verschillende soorten display devices, die bij computer graphics gebruikt kunnen worden. Onder display files komen o.a. aan de orde display file compilers, 2-dim. transformaties, clipping en windowing. Ook worden transformatie systemen behandeld. Bij interactieve graphics wordt aandacht besteed aan grafische input devices (b.v. tablets, lightpen), interrupthandling (attention queue) en interactieve grafische technieken. In het geval van 3-dim. computer graphics wordt ingegaan op projecties, al of niet met perspectief, van 3-dim. objecten (3-dim. transformaties). Bewerkingen zoals hidden line elimination en shading besluiten dit gedeelte. Het laatste hoofdstuk, grafische systemen, behandelt command languages, programmeertalen voor computer graphics en het ontwerpen van grafische systemen. Een laatste opmerking over dit boek is, dat het een goede basiskennis geeft over computer graphics.

Het tweede boek is 'Computer graphics' van J.L. Encarnacao, die hiermee iets dieper ingaat op de verschillende problemen vergeleken bij Sproull/Newman. De belangrijkste onderwerpen in het boek zijn:

- computer graphics systemen (H.2)
- datastructuren (H.3)
- grafische software (H.4)
- wiskundige berekeningen voor computer graphics (H.5)
- speciale problemen bij 3-dim. weergave (H.6).

De datastructuren worden aan de hand van enkele voorbeelden behandeld (CORAL, LEAP, ASP, APL, DATAS). Ook wordt behandeld hoe

met behulp van de 'graphen theorie' datastructuren kunnen worden ontworpen en beschreven (R. Williams) en tenslotte volgt nog een beschrijving van DSPS (Data Structure Programming System). Ook in H.4 worden voorbeelden aangehaald om de grafische software te behandelen. Van de interactieve programmeertalen zijn dit GRAF, GPL/1, APLG, EX.GRAF. Verder bevinden zich nog interessante onderwerpen in H.5 en H.6, zoals coördinaten-transformatie, perspectieve afbeelding, het weergeven van krommen en gewelfde vlakken, windowing, zichtbaarheids-criteria, shaded pictures.

Het boek 'Graphic languages' van Nake/Rosenfeld bestaat uit een aantal lezingen, waarvan enkele hieronder worden opgenoemd:

- Newman, W.M. blz. 291
  A prototype low cost single-user graphics system
- Boullier, P. blz. 244
  Metavisu, a general purpose graphic system
- Williams, R. blz. 334
  A general purpose graphical language
- Takasawa/Moriguchi blz. 327
  A graphics manipulating language (GML)

Ook van de onder [6] genoemde literatuur kan gezegd worden dat er zich interessante artikelen onder bevinden. Enkele voorbeelden hiervan zijn:

uit Computer Graphics ACM (U.W. Pooch):

- Thanhouser blz. 13
  Intermixing refresh and direct view storage graphics
- Pooch blz. 25
  A user oriented computer graphics system (CGS)
- Bergman/Kaufman blz. 133
  BGRAF2: a real time graphics language with modular objects and implicit dynamics
- Schrack blz. 173
  On the semantics of the assignment statement of high-level graphical languages
- Waller/William blz. 183
  Graphic and relational data base support for problem solving

- Moulton/Corman blz. 204
  Remote programmability of graphic interactions in a host/satelite configuration

uit ACM Siggraph/Sigplan (T. Berk):

- Schrack blz. 10
  Design, implementation and experiences with a higher level graphics language for interactive computer aided design purposes (LIG)
- Jones blz. 18
  An extended ALGOL-60 for shaded computer graphics

uit Computer Graphics ACM (A.P. Lucido):

- Koenigsberg/Meads/Shaw/Thanhouser/Vollum blz. 42
  A graphics operating system
- Giloi, W.K. blz. 61
  On high level programming systems for structured display programming

APPENDIX A.

BESCHRIJVING VAN ENKELE GRAFISCHE PAKKETTEN

Om inzicht te geven in het onderzoek, dat in Nederland op het gebied van computer graphics is en nog wordt verricht, zijn een aantal resultaten beschreven, die hieruit hun bestaan te danken hebben.
Daarna wordt in deze appendix een korte beschrijving gegeven van het pakket GINO-F.
Als laatste wordt melding gemaakt van de werkzaamheden van het Graphics Standards Planning Committee om tot een internationale standaardisering op het gebied van computer graphics te komen.

A.1. Onderzoek in Nederland op het gebied van computer graphics

Onderzoek op het gebied van computer graphics heeft in Nederland tot de volgende resultaten geleid:

- GPGS

  De TH-Delft en de universiteiten van Nijmegen en Cambridge hebben GPGS (General Purpose Graphic System) ontwikkeld. GPGS kan beschouwd worden als een device onafhankelijk subroutine-pakket, dat zijn interactieve en passieve grafische mogelijkheden d.m.v. het subroutine call mechanisme van hogere programmeertalen gemakkelijk hieraan laat toevoegen.
  Bij het ontwerpen heeft men getracht het systeem zo eenvoudig en zo algemeen mogelijk te houden. Hiertoe zijn toepassingen aanwezig zoals 2- en 3-dim. windowing, clipping en transformaties.
  Wat betreft de device onafhankelijkheid onderscheid men bij elke implementatie een device onafhankelijk deel, dat door het applicatie-programma kan worden aangeroepen, en device afhankelijke device drivers. Men dient bij het programmeren in gedachte te hebben, dat men bezig is voor een enkel geïdealiseerd device.
  De gang van zaken bij het opbouwen van een display file is als volgt. Het applicatie programma bouwt een picture op d.m.v. subroutine calls aan GPGS. Hierdoor wordt een display file of een picture programma voor de display processor unit (DPU) van het display device gemaakt door de GPGS software. De display file wordt opgebouwd als een verzameling van picture segmenten, dit zijn de kleinste eenheden, die voor manipulatie in aanmerking komen (create, extend, delete). Deze picture segmenten worden weer opgebouwd uit picture elementen, zoals lijnstukken, character strings en cirkels en hun attributen. In GPGS is verder ook nog sprake van device onafhankelijke pseudo picture segmenten.
  Literatuur:

  GPGS General purpose graphic system L.C. Caruthers

- PHILDIG

  Philips is bezig met de ontwikkeling van PHILDIG (Philips Device Independent Graphics). Het pakket bestaat uit een bovenlaag van

randapparaat onafhankelijke routines en voor elk apparaat een aantal speciale routines met een voor dat apparaat specifiek common block. In de literatuur wordt hoofdzakelijk ingegaan op een aantal soorten CALLS (voor administratie, omlijsting, generatie, structurering, transformatie, invoer en fout-afhandeling).

Literatuur:

PHILDIG door ir. C. Niessen/ir. J.W. Ero.

- ILP

In het Mathematisch Centrum in Amsterdam is men bezig met een onderzoek voor een intermediate language for pictures (ILP). De hoofdlijnen hiervan zijn te vinden in de volgende literatuur:

- Colloquium structuur van programmeertalen
  MC syllabus 25 1976
  Blz. 23-40: Grafische programmeertalen, P.J.W. ten Hagen.
- An introduction to Computer Graphics, P. Klint.
- The Intermediate language for pictures, P.J.W. ten Hagen.

In 'An introduction to computer graphics' van P. Klint komen aan de orde:

1. Graphics devices.
2. Graphics System Organisation.
   Dit hoofdstuk geeft een beschrijving, zoals die ongeveer te vinden is in Sproull/Newman: Principles of interactive computer graphics.
   Hij behandelt twee mechanismen voor het aanbrengen van veranderingen aan pictures, die reeds zijn gegenereerd. Dit zijn de 'viewing algoritme' met gebruik van een Structured Picture Description (SPD) en de 'Plotter Analogy' met gebruik van een transformed display file (TDF).
3. Een model voor grafische I/O funktie.
   Hier wordt o.a. een device description table (DDT) ingevoerd om de relatie tussen primitives en hardware mogelijkheden van een specifiek device weer te geven. De DDT kan dus beschouwd worden als de interface tussen SPD en grafische I/O devices.

P.J.W. ten Hagen behandelt ILP (Intermediate language for pictures).

De ILP wordt op de volgende manieren toegepast:

- Alle pictures worden gerepresenteerd als ILP programma's.
- De communicatie tussen gebruiker en het applicatie-programma gebeurt in termen van ILP programma's.
- De high-level grafische taal van het applicatie-programma wordt verkregen door ILP in een bestaande high-level general purpose programmataal in te bedden.
- Pictures kunnen worden opgeslagen in geheugen, aangeroepen en geklassificeerd worden als ILP programma's.
- De verscheidene tekenmachines zijn logisch verbonden door het definiëren van een omzetting tussen ILP en machine-code. Dit is ook het geval voor input devices. De belangrijke consequentie van de laatste toepassing is dat voor het eerst volledige symmetrie tussen input en output kan worden verkregen.

A.2. GINO-F

Het GINO-F (Graphical Input/Output - Fortran version) general purpose graphics package is ontworpen op het Computer Aided Design Centre in Cambridge. Dit grafische pakket heeft de vorm van een bibliotheek van teken- en administratieve subroutines, die grotendeels zijn geschreven in Standard ANSI FORTRAN. GINO-F is device onafhankelijk. Er zijn hiervoor routines (code generators) aanwezig, die uitvoerdata kunnen produceren voor verschillende devices. De mogelijkheden van GINO-F variëren van het produceren van grafische output voor 2-dim. graphs tot complexe 3-dim. interactieve systemen. Men vindt in GINO-F voorzieningen voor:

- Het tekenen van lijnstukken of van gedeelten van cirkels, waarbij voor het type lijn gekozen kan worden uit solid, dashed of chained.
- De uitvoer van characters en getallen in verschillende vormen, punten en een aantal symbolen.
- Het selecteren en vrijgeven van devices, het kiezen van papier en pen afmetingen en types.
- Het uitvoeren van (viewing) transformations en windowing op object definities, waarbij dan het effect hiervan gecontroleerd kan worden.
- Het opbouwen van picture libraries.
- Interactie voor zowel refresh als non-refresh displays.

A.3. Siggraph's Graphics Standards Planning Committee

Het doel van de GSPC is het definiëren van de functionele specificaties van een 'line drawing graphics programming system', dat voor zowel output als input device-onafhankelijk is. Het is verder de bedoeling dat een systeem, dat aan deze specificaties tegemoet komt, kan worden geïmplementeerd als een subroutine-pakket, dat aanroepbaar is door talen zoals FORTRAN en PL/1. Dit alles om een internationale standaardisering op het gebied van computer graphics te bereiken. Er zijn GSPC subgroepen met de volgende opdrachten:

- De functionele specificaties voor een binnenste kern (CORE) van het gehele systeem te definiëren, dat voorziet in het verwerken van zowel input als output.
- Bestaande pakketten aan een onderzoek te onderwerpen.
- Het definiëren van de interface tussen de CORE en higher level software (modelling system) en tussen de CORE en lower level software (operating system).

Aan de orde komen nu de werkzaamheden van groep 1, zoals die in het eerste verslag van deze groep beschreven zijn (oct. 1976). Hierin wordt onderscheid gemaakt tussen de volgende twee systemen:

- Modelling system

  In dit systeem worden objecten gedefinieerd in hun eigen locale coördinaten. Het systeem bevat functies voor het samenstellen en het combineren van transformaties en deze transformaties uit te voeren op locale coördinaten. Het doel van de transformaties is het produceren van een definitie van de weer te geven informatie in een 'wereld'-coördinaten-stelsel, geschikt voor toepassing van de viewing transformatie(s).

  Functies in dit systeem:

  - het samenstellen van transformaties.
  - het plaatsen van transformaties op een transformatie stack; transformaties van deze stack afhalen.
  - het uitvoeren van transformaties op punten, die in locale coördinaten gedefinieerd zijn.

- (CORE) graphics system
  M.b.v. dit systeem wordt de viewing transformatie(s) opgezet en wordt de 'picture' gegenereerd. Dit gebeurt door het aanroepen van 'line drawing functions' met 'wereld'-coördinaten argumenten.
  Functies in dit systeem:
  - primitieve functies voor het tekenen van lijnen en tekst.
  - attributen functies.
  - viewing transformatie(s).
  - input functies.
  - functies voor o.a. segmentatie, initialisatie.

Een interactief grafisch systeem kan worden voorgesteld door een 'pipeline', waarin de volgende datastructuren en operaties voorkomen:

- De applicatie programmeur ontwerpt een Application Data Structure (ADS), die geometrische en applicatie georiënteerde data bevat.
- Een vertaler onttrekt van de ADS geometrische informatie voor object-delen en gebruikt het modelling subsysteem om een 2-dim. of 3-dim.'wire frame object' op te bouwen. De locale coördinaten worden verder getransformeerd naar device onafhankelijke 'wereld'-coördinaten.
- Een stroom van 'wereld'-coördinaten wordt doorgegeven aan de CORE, die een device afhankelijke 'picture' van het object maakt (met viewing transformation). De device afhankelijke 'picture' kan worden gesegmenteerd om statische van dynamische informatie af te zonderen (identificatie).
- Operator interaction voor het veranderen van b.v.
  - viewing parameters (CORE)
  - geometrische relaties van object-delen in het totale object (modelling system)
  - ADS

```
100 PROGRAM OBJECTPROC (INPUT, OUTPUT);
200 CONST DETDEF = FALSE;
300       PENDEF = 70;
400       COLDEF = 0;
500       DASH1DEF = 100;
600       GPEYDEF =0;
700       ITALDEF = 0;
800       CHARWDEF = 0;
900       CHARHDEF = 0;
1000       NMAX = 10;
1100       PI = 3.14159265359;
1200
1300 TYPE   POINTL = @ELEMENT;
1400        LINEL = @ELEMENT;
1500        CONTOURL = @ELEMENT;
1600        TEXTL = @ELEMENT;
1700        OBJELL = @ELEMENT;
1800        EPTR = @ELEMENT;
1900        OPTR = @OBJECT;
2000        ONAME = ARRAY[1..10] OF CHAR;
2100        MATRIX = ARRAY[1..4,1..4] OF REAL;
2200        OBJECT = RECORD
2300                  PLINK : POINTL;
2400                  LLINK : LINEL;
2500                  CLINK : CONTOURL;
2600                  TLINK : TEXTL;
2700                  OELINK : OBJELL
2800                 END;
2900        OBJDESCR = RECORD
3000                    OBJ : ONAME;
3100                    OBJLOC : OPTR;
3200                    NOFREF : INTEGER
3300                   END;
3400        ELEM = (POINTE, LINEE, CONTOURE, TEXTE, OBJE);
3500        LDESCR = RECORD
3600                  PERIOD : REAL;
3700                  DASH1, GAP, DASH2 : 0..100
3800                 END;
3900        COORDTRP = @COORDTR;
4000        COORDTR = RECORD
4100                    FLINK : COORDTRP;
4200                    X, Y, Z : REAL
4300                   END;
4400        REPR = RECORD
4500                RELDET, RELPEN, RELCOL, RELLS, RELGREY : BOOLEAN;
4600                DETECT : BOOLEAN;
4700                PEN : -100..100;
4800                COLOUR : -100..100;
4900                CASE ELEMENTTYPE : ELEM OF
5000                       LINEE : (LLST : LDESCR);
5100                       CONTOURE : (CLST : LDESCR;
5200                                   CGREY : -100..100);
5300                       TEXTE : (ITALIC, CHARW, CHARH : REAL);
5400                       OBJE : (OLST : LDESCR;
5500                               OGREY : -100..100)
5600               END;
#
```

```
                    FLINK : @ELEMENT;
                    BLINK : @ELEMENT;
                    RALINK : @REPR;
                    CASE ELEMENTTYPE : ELEM OF
                    POINTE : (X, Y, Z : REAL);
                    LINEE : (XB, YB, ZB, XE, YE, ZE : REAL);
                    CONTOURE : (COORDTRL : COORDTRP;
                                NOFCOORDTRS : INTEGER);
                    TEXTE : (XLO, YLO, ZLO, XLB, YLB, ZLB,
                             XRO, YRO, ZRO : REAL;
                             CHARS : ARRAY[1..10] OF CHAR);
                    OBJE : (TMATRIX : MATRIX;
                            OBJREF : ONAME;
                            INDEX : INTEGER)
                    END;
          POINTIT = (X, Y, Z, PDET, PPEN, PCOL);
          LINEIT = (XB, YB, ZB, XE, YE, ZE, LDET, LPEN, LCOL,
                    LLST);
           CONTOURIT = (COORDTRS, CDET, CPEN, CCOL, CLST, CGREY);
          TEXTIT = (XLO, YLO, ZLO, XLB, YLB, ZLB, XRO, YRO, ZRO,
                    CHARS, TDET, TPEN, TCOL, ITALIC, CHARW, CHARH);
          OBJIT = (OBJREF, TMATRIX, ODET, OPEN, OCOL, OLST, OGREY);
          ELEMINF = RECORD
                    LASTREF : @ELEMENT;
                    OBJ : ONAME;
                    RELDET, RELPEN, RELCOL, RELLS, RELGREY : BOOLEAN;
                    DETECT : BOOLEAN;
                    PEN, COLOUR : -100..100;
                    CASE ELEMENTTYPE : ELEM OF
                    POINTE    : (PVAL : SET OF POINTIT;
                                 X, Y, Z : REAL);
                    LINEE     : (LVAL : SET OF LINEIT;
                                 XB, YB, ZB, XE, YE, ZE : REAL;
                                 LLST : LDESCR);
                    CONTOURE : (CVAL : SET OF CONTOURIT;
                                COORDTRL : COORDTRP;
                                NOFCOORDTRS : INTEGER;
                                CLST : LDESCR;
                                CGREY : -100..100);
                    TEXTE     : (TVAL : SET OF TEXTIT;
                                 XLO, YLO, ZLO, XLB, YLB, ZLB,
                                 XRO, YRO, ZRO : REAL;
                                  CHARS : ARRAY[1..10] OF CHAR;
                                  ITALIC, CHARW, CHARH : REAL);
                     OBJE      : (OEVAL : SET OF OBJIT;
                                  OBJREF : ONAME;
                                  TMATRIX : MATRIX;
                                  OLST : LDESCR;
                                  OGREY : -100..100)
                    END;

VAR     OBJADM : ARRAY[1..NMAX] OF OBJDESCR;
        E : ELEMINF;
        OBJ : ONAME;
        ITEM, OPERATION, I : INTEGER;
        ENDOFD, ENDOFPR : BOOLEAN;
        TM : MATRIX;
        CH, ICH, ELTYPE : CHAR;
        OP, OP1, OP2 : OPTR;
        LST : LDESCR;
        VX, VY, VZ, DX, DY, DZ : REAL;
*
```

```
12000 PROCEDURE NEXTEL (VAR EP : EPTR; OP : OPTR);
12100  (* IF EP=NIL: EP BECOMES THE POINTER TO THE FIRST ELEMENT  *)
12200  (*             IN OBJECT OP@.                              *)
12300  (* IF EP<>NIL: EP BECOMES THE POINTER TO THE NEXT ELEMENT  *)
12400  (*             IN OBJECT OP@ AFTER ELEMENT EP@; HOWEVER    *)
12500  (*             WHEN EP@ WAS THE LAST ELEMENT THEN EP:=NIL  *)
12600 VAR   ETYPE : ELEM;
12700 BEGIN IF OP<>NIL THEN
12800       BEGIN
12900       IF EP=NIL THEN
13000       BEGIN EP:=OP@.PLINK;
13100             IF EP=NIL THEN
13200             BEGIN EP:=OP@.LLINK;
13300                   IF EP=NIL THEN
13400                   BEGIN EP:=OP@.CLINK;
13500                         IF EP=NIL THEN
13600                         BEGIN EP:=OP@.TLINK;
13700                               IF EP=NIL THEN EP:=OP@.OELINK
13800                         END
13900                   END
14000             END
14100       END ELSE
14200       BEGIN ETYPE:=EP@.ELEMENTTYPE;
14300             EP:=EP@.FLINK;
14400             WHILE (EP=NIL) AND (ETYPE<>OBJE) DO
14500                   CASE ETYPE OF
14600                   POINTE    : BEGIN EP:=OP@.LLINK;
14700                                     ETYPE:=LINEE
14800                                 END;
14900                   LINEE     : BEGIN EP:=OP@.CLINK;
15000                                     ETYPE:=CONTOURE
15100                                 END;
15200                   CONTOURE  : BEGIN EP:=OP@.TLINK;
15300                                     ETYPE:=TEXTE
15400                                 END;
15500                   TEXTE     : BEGIN EP:=OP@.OELINK;
15600                                     ETYPE:=OBJE
15700                                 END
15800                   END
15900       END
16000       END
16100 END;
16200
16300 FUNCTION GETIND (OBJ : ONAME) : BOOLEAN;
16400  (* I BECOMES THE INDEX-VALUE OF THE OBJADM-RECORD OF       *)
16500  (* OBJECT OBJ                                              *)
16600 BEGIN I:=1;
16700       WHILE (I<>NMAX) AND (OBJADM[I].OBJ<>OBJ) DO I:=I+1;
16800       IF OBJADM[I].OBJ=OBJ THEN GETIND:=TRUE
16900                            ELSE GETIND:=FALSE
17000 END;
      #
```

```
17200 PROCEDURE RETSPACE (OP : OPTR);
17300  (* THE STORAGE OF ALL ELEMENTS OF OBJECT OP@ IS GIVEN FREE.*)
17400 VAR   DEP, EP : EPTR;
17500       CTRP, DCTRP : COORDTRP;
17600 BEGIN EP:=NIL;
17700       NEXTEL(EP, OP);
17800       WHILE EP<>NIL DO
17900       BEGIN DEP:=EP;
18000             NEXTEL(EP, OP);
18100             CASE DEP@.ELEMENTTYPE OF
18200  (*         POINTE :                                          *)
18300  (*          BEGIN                                            *)
18400  (*           IF DEP@.RALINK<>NIL THEN                        *)
18500  (*               DISPOSE(DEP@.RALINK, POINTE);               *)
18600  (*           DISPOSE(DEP, POINTE)                            *)
18700  (*          END;                                             *)
18800  (*         LINEE :                                           *)
18900  (*          BEGIN                                            *)
19000  (*           IF DEP@.RALINK<>NIL THEN                        *)
19100  (*               DISPOSE(DEP@.RALINK, LINEE);                *)
19200  (*           DISPOSE(DEP, LINEE)                             *)
19300  (*          END;                                             *)
19400  (*         CONTOURE :                                        *)
19500  (*         BEGIN CTRP:=DEP@.COORDTRL;                        *)
19600  (*               WHILE CTRP<>NIL DO                          *)
19700  (*               BEGIN DCTRP:=CTRP;                          *)
19800  (*                     CTRP:=DCTRP@.FLINK;                   *)
19900  (*                     DISPOSE(DCTRP)                        *)
20000  (*               END;                                        *)
20100  (*               IF DEP@.RALINK<>NIL THEN                    *)
20200  (*                    DISPOSE(DEP@.RALINK, CONTOURE);        *)
20300  (*               DISPOSE(DEP, CONTOURE)                      *)
20400  (*         END;                                              *)
20500  (*         TEXTE :                                           *)
20600  (*          BEGIN                                            *)
20700  (*           IF DEP@.RALINK<>NIL THEN                        *)
20800  (*                DISPOSE(DEP@.RALINK, TEXTE);               *)
20900  (*           DISPOSE(DEP, TEXTE)                             *)
21000  (*          END;                                             *)
21100             OBJE :
21200             BEGIN OBJADM[DEP@.INDEX].NOFREF:=
21300                        OBJADM[DEP@.INDEX].NOFREF-1;
21400                   IF (OBJADM[DEP@.INDEX].NOFREF=0) AND
21500                      (OBJADM[DEP@.INDEX].OBJ='NO OBJECT ') THEN
21600                   BEGIN RETSPACE(OBJADM[DEP@.INDEX].OBJLOC);
21700                         OBJADM[DEP@.INDEX].OBJLOC:=NIL
21800                   END;
21900  (*               IF DEP@.RALINK<>NIL THEN                    *)
22000  (*                  DISPOSE(DEP@.RALINK, OBJE);              *)
22100  (*               DISPOSE(DEP, OBJE)                          *)
22200             END
22300             END
22400       END
22500 END;
      #
```

```
22700 PROCEDURE ADDE (EP : EPTR; OP : OPTR);
22800  (* THE ELEMENT EP@ WILL BE PLACED IN OBJECT OP@ AS THE     *)
22900  (* FIRST ELEMENT IN THE COLLECTION OF ITS TYPE.            *)
23000 BEGIN EP@.BLINK:=NIL;
23100       CASE EP@.ELEMENTTYPE OF
23200       POINTE :    BEGIN IF OP@.PLINK<>NIL
23300                         THEN BEGIN OP@.PLINK@.BLINK:=EP;
23400                                    EP@.FLINK:=OP@.PLINK
23500                              END
23600                         ELSE EP@.FLINK:=NIL;
23700                         OP@.PLINK:=EP
23800                   END;
23900       LINEE :     BEGIN IF OP@.LLINK<>NIL
24000                         THEN BEGIN OP@.LLINK@.BLINK:=EP;
24100                                    EP@.FLINK:=OP@.LLINK
24200                              END
24300                         ELSE EP@.FLINK:=NIL;
24400                         OP@.LLINK:=EP;
24500                   END;
24600       CONTOURE : BEGIN IF OP@.CLINK<>NIL
24700                         THEN BEGIN OP@.CLINK@.BLINK:=EP;
24800                                    EP@.FLINK:=OP@.CLINK
24900                              END
25000                         ELSE EP@.FLINK:=NIL;
25100                         OP@.CLINK:=EP;
25200                   END;
25300       TEXTE :     BEGIN IF OP@.TLINK<>NIL
25400                         THEN BEGIN OP@.TLINK@.BLINK:=EP;
25500                                    EP@.FLINK:=OP@.TLINK
25600                              END
25700                         ELSE EP@.FLINK:=NIL;
25800                         OP@.TLINK:=EP;
25900                   END;
26000       OBJE :      BEGIN IF OP@.OELINK<>NIL
26100                         THEN BEGIN OP@.OELINK@.BLINK:=EP;
26200                                    EP@.FLINK:=OP@.OELINK
26300                              END
26400                         ELSE EP@.FLINK:=NIL;
26500                         OP@.OELINK:=EP;
26600                         OBJADM[EP@.INDEX].NOFREF:=
26700                         OBJADM[EP@.INDEX].NOFREF+1
26800                   END
26900       END
27000 END;
27100
27200 FUNCTION LOCOBJ (OBJ : ONAME; VAR OP : OPTR) : BOOLEAN;
27300  (* OP BECOMES THE POINTER TO THE OBJECT-DEFINITION OF 'OBJ'*)
27400 BEGIN IF GETIND(OBJ) THEN
27500       BEGIN OP:=OBJADM[I].OBJLOC;
27600             LOCOBJ:=TRUE
27700       END ELSE LOCOBJ:=FALSE
27800 END;
      #
```

```
28000 PROCEDURE ADMLOC (OBJ : ONAME; OP : OPTR);
28100  (* THE POINTER OP IS REGISTRATED IN THE OBJADM-RECORD OF   *)
28200  (* OBJECT OBJ                                             *)
28300 BEGIN IF GETIND(OBJ) THEN
28400       BEGIN IF OBJADM[I].OBJLOC<>NIL THEN
28500             BEGIN RETSPACE(OBJADM[I].OBJLOC);
28600                   (* DISPOSE(OBJADM[I].OBJLOC) *)
28700             END;
28800             OBJADM[I].OBJLOC:=OP
28900       END (* ELSE OBJ NOT DECLARED *)
29000 END;
29100
29200 PROCEDURE COPYCTRS (NEWE : EPTR; CTRP : COORDTRP);
29300  (* THE COORD.-TRIPLES OF CONTOUR-ELEMENT EP@ ARE          *)
29400  (* COPIED AND ASSIGNED TO CONTOUR-ELEMENT NEWE@.          *)
29500 VAR   NEWCTR1, NEWCTR2 : COORDTRP;
29600 BEGIN NEW(NEWCTR1);
29700       NEWE@.COORDTRL:=NEWCTR1;
29800       NEWCTR1@:=CTRP@;
29900       CTRP:=CTRP@.FLINK;
30000       WHILE CTRP<>NIL DO
30100       BEGIN NEWCTR2:=NEWCTR1;
30200             NEW(NEWCTR1);
30300             NEWCTR2@.FLINK:=NEWCTR1;
30400             NEWCTR1@:=CTRP@;
30500             CTRP:=CTRP@.FLINK
30600       END
30700 END;
30800
30900 PROCEDURE COPYOBJ (OP, NEWO : OPTR);
31000  (* COPIES OF ALL ELEMENTS IN OBJECT OP@ ARE ADDED TO       *)
31100  (* OBJECT NEWO@.                                          *)
31200 VAR   NEWE, EP : EPTR;
31300       NEWA : @REPR;
31400 BEGIN EP:=NIL;
31500       IF OP<>NIL THEN
31600       BEGIN
31700       NEXTEL(EP, OP);
31800       WHILE EP<>NIL DO
31900       BEGIN CASE EP@.ELEMENTTYPE OF
32000             POINTE : NEW(NEWE, POINTE);
32100             LINEE : NEW(NEWE, LINEE);
32200             CONTOURE : NEW(NEWE, CONTOURE);
32300             TEXTE : NEW(NEWE, TEXTE);
32400             OBJE : NEW(NEWE, OBJE)
32500             END;
32600             NEWE@:=EP@;
32700             IF NEWE@.RALINK<>NIL THEN
32800             BEGIN CASE NEWE@.ELEMENTTYPE OF
32900                   POINTE : NEW(NEWA, POINTE);
33000                   LINEE : NEW(NEWA, LINEE);
33100                   CONTOURE : NEW(NEWA, CONTOURE);
33200                   TEXTE : NEW(NEWA, TEXTE);
33300                   OBJE : NEW(NEWA, OBJE)
33400                   END;
33500                   NEWA@:=EP@.RALINK@;
33600                   NEWE@.RALINK:=NEWA
33700             END;
          #
```

```
                IF EP@.ELEMENTTYPE=CONTOURE THEN
                   COPYCTRS(NEWE, EP@.COORDTRL);
                ADDE(NEWE,NEWO);
                NEXTEL(EP,OP)
          END
          END
END;

PROCEDURE DELE (EP : EPTR; OP : OPTR);
 (* ELEMENT EP@ WILL BE DELETED FROM OBJECT OP@.                    *)
VAR   CTRP, DCTRP : COORDTRP;
BEGIN IF EP@.BLINK<>NIL
      THEN EP@.BLINK@.FLINK:=EP@.FLINK
      ELSE CASE EP@.ELEMENTTYPE OF
           POINTE    : OP@.PLINK:=EP@.FLINK;
           LINEE     : OP@.LLINK:=EP@.FLINK;
           CONTOURE  : OP@.CLINK:=EP@.FLINK;
           TEXTE     : OP@.TLINK:=EP@.FLINK;
           OBJE      : OP@.OELINK:=EP@.FLINK
           END;
      IF EP@.FLINK<>NIL THEN EP@.FLINK@.BLINK:=EP@.BLINK;
      CASE EP@.ELEMENTTYPE OF
(*    POINTE :                                                        *)
(*     BEGIN                                                          *)
(*      IF EP@.RALINK<>NIL THEN DISPOSE(EP@.RALINK, POINTE);          *)
(*      DISPOSE(EP, POINTE)                                           *)
(*     END;                                                           *)
(*    LINEE :                                                         *)
(*     BEGIN                                                          *)
(*      IF EP@.RALINK<>NIL THEN DISPOSE(EP@.RALINK, LINEE);           *)
(*      DISPOSE(EP, LINEE)                                            *)
(*     END;                                                           *)
(*    CONTOURE :                                                      *)
(*    BEGIN CTRP:=EP@.COORDTRL;                                       *)
(*          WHILE CTRP<>NIL DO                                        *)
(*          BEGIN DCTRP:=CTRP;                                        *)
(*                CTRP:=DCTRP@.FLINK;                                 *)
(*                DISPOSE(DCTRP)                                      *)
(*          END;                                                      *)
(*          IF EP@.RALINK<>NIL THEN                                   *)
(*               DISPOSE(EP@.RALINK, CONTOURE);                       *)
(*          DISPOSE(EP, CONTOURE)                                     *)
(*    END;                                                            *)
(*    TEXTE :                                                         *)
(*     BEGIN                                                          *)
(*      IF EP@.RALINK<>NIL THEN DISPOSE(EP@.RALINK, TEXTE);           *)
(*      DISPOSE(EP, TEXTE)                                            *)
(*     END;                                                           *)
      OBJE :
      BEGIN OBJADM[EP@.INDEX].NOFREF:=
            OBJADM[EP@.INDEX].NOFREF-1;
            IF (OBJADM[EP@.INDEX].NOFREF=0) AND
               (OBJADM[EP@.INDEX].OBJ='NO OBJECT ') THEN
            BEGIN RETSPACE(OBJADM[EP@.INDEX].OBJLOC);
                  OBJADM[EP@.INDEX].OBJLOC:=NIL
            END;
(*          IF EP@.RALINK<>NIL THEN                                   *)
(*               DISPOSE(EP@.RALINK, OBJE);                           *)
(*          DISPOSE(EP, OBJE)                                         *)
      END
      END
END;
#
```

```
40100 FUNCTION CMPIT (E : ELEMINF; EP : EPTR) : BOOLEAN;
40200  (* THIS FUNCTION BECOMES TRUE, IF,  FOR THE ITEMS DEFINED   *)
40300  (* IN E. VAL, ELEMENT EP@ IS EQUAL TO THE ELEMENT DESCRIBED *)
40400  (* IN THE VARIABLE 'E'.                                     *)
40500 VAR   ITP : POINTIT;
40600       ITL : LINEIT;
40700       ITC : CONTOURIT;
40800       ITT : TEXTIT;
40900       ITOE : OBJIT;
41000       IDENT : BOOLEAN;
41100       ECTRP, EPCTRP : COORDTRP;
41200 BEGIN IF EP<>NIL THEN
41300       BEGIN
41400       IDENT:=TRUE;
41500       CASE E.ELEMENTTYPE OF
41600       POINTE :
41700        BEGIN
41800         IF (EP@.RALINK=NIL) AND
41900            (E.PVAL*[PDET, PPEN, PCOL]<>[])
42000         THEN IDENT:=FALSE ELSE
42100         BEGIN
42200         FOR ITP:=X TO PCOL DO
42300         IF IDENT THEN
42400         BEGIN IF ITP IN E.PVAL THEN
42500               CASE ITP OF
42600               X : IF E.X<>EP@.X THEN IDENT:=FALSE;
42700               Y : IF E.Y<>EP@.Y THEN IDENT:=FALSE;
42800               Z : IF E.Z<>EP@.Z THEN IDENT:=FALSE;
42900               PDET : IF E.DETECT<>EP@.RALINK@.DETECT THEN
43000                                    IDENT:=FALSE;
43100               PPEN : IF E.PEN<>EP@.RALINK@.PEN THEN
43200                                    IDENT:=FALSE;
43300               PCOL : IF E.COLOUR<>EP@.RALINK@.COLOUR THEN
43400                                    IDENT:=FALSE
43500               END
43600         END
43700         END
43800        END;
43900       LINEE :
44000        BEGIN
44100         IF (EP@.RALINK=NIL) AND
44200            (E.LVAL*[LDET, LPEN, LCOL, LLST]<>[])
44300         THEN IDENT:=FALSE ELSE
44400         BEGIN
44500         FOR ITL:=XB TO LLST DO
44600         IF IDENT THEN
44700         BEGIN IF ITL IN E.LVAL THEN
44800               CASE ITL OF
44900               XB : IF E.XB<>EP@.XB THEN IDENT:=FALSE;
45000               YB : IF E.YB<>EP@.YB THEN IDENT:=FALSE;
45100               ZB : IF E.ZB<>EP@.ZB THEN IDENT:=FALSE;
45200               XE : IF E.XE<>EP@.XE THEN IDENT:=FALSE;
45300               YE : IF E.YE<>EP@.YE THEN IDENT:=FALSE;
45400               ZE : IF E.ZE<>EP@.ZE THEN IDENT:=FALSE;
45500               LDET : IF E.DETECT<>EP@.RALINK@.DETECT THEN
45600                                    IDENT:=FALSE;
45700               LPEN : IF E.PEN<>EP@.RALINK@.PEN THEN
45800                                    IDENT:=FALSE;
45900               LCOL : IF E.COLOUR<>EP@.RALINK@.COLOUR THEN
46000                                    IDENT:=FALSE;
            #
```

```
46100              LLST : IF E.LLST<>EP@.RALINK@.LLST THEN
46200                                        IDENT:=FALSE
46300              END
46400           END
46500           END
46600          END;
46700         CONTOURE :
46800          BEGIN
46900           IF (EP@.RALINK=NIL) AND
47000              (E.CVAL*[CDET, CPEN, CCOL, CLST, CGREY]<>[])
47100           THEN IDENT:=FALSE ELSE
47200           BEGIN
47300           FOR ITC:=COORDTRS TO CSHAD DO
47400           IF IDENT THEN
47500           BEGIN IF ITC IN E.CVAL THEN
47600                 CASE ITC OF
47700                 COORDTRS : BEGIN
47800                              IF E.NOFCOORDTRS<>EP@.NOFCOORDTRS
47900                              THEN IDENT:=FALSE ELSE
48000                              BEGIN
48100                               ECTRP:=E.COORDTRL;
48200                               EPCTRP:=EP@.COORDTRL;
48300                               WHILE (ECTRP<>NIL) AND IDENT DO
48400                               BEGIN
48500                                IF (ECTRP@.X=EPCTRP@.X) AND
48600                                   (ECTRP@.Y=EPCTRP@.Y) AND
48700                                   (ECTRP@.Z=EPCTRP@.Z) THEN
48800                                BEGIN ECTRP:=ECTRP@.FLINK;
48900                                      EPCTRP:=EPCTRP@.FLINK
49000                                END ELSE IDENT:=FALSE
49100                               END
49200                              END;
49300                            END;
49400                 CDET : IF E.DETECT<>EP@.RALINK@.DETECT THEN
49500                                        IDENT:=FALSE;
49600                 CPEN : IF E.PEN<>EP@.RALINK@.PEN THEN
49700                                        IDENT:=FALSE;
49800                 CCOL : IF E.COLOUR<>EP@.RALINK@.COLOUR THEN
49900                                        IDENT:=FALSE;
50000                 CLST : IF E.CLST<>EP@.RALINK@.CLST THEN
50100                                        IDENT:=FALSE;
50200                 CGREY : IF E.CGREY<>EP@.RALINK@.CGREY THEN
50300                                        IDENT:=FALSE
50400                 END
50500           END
50600           END
50700          END;
50800         TEXTE :
50900          BEGIN
51000           IF (EP@.RALINK=NIL) AND
51100              (E.TVAL*[TDET, TPEN, TCOL, ITALIC, CHARW, CHARH]
51200                                     <>[])
51300           THEN IDENT:=FALSE ELSE
51400           BEGIN
51500           FOR ITT:=XLO TO CHARH DO
51600           IF IDENT THEN
51700           BEGIN IF ITT IN E.TVAL THEN
51800                 CASE ITT OF
51900                 XLO : IF E.XLO<>EP@.XLO THEN IDENT:=FALSE;
52000                 YLO : IF E.YLO<>EP@.YLO THEN IDENT:=FALSE;
                      #
```

```
52100              ZLO : IF E.ZLO<>EP@.ZLO THEN IDENT:=FALSE;
52200              XLB : IF E.XLB<>EP@.XLB THEN IDENT:=FALSE;
52300              YLB : IF E.YLB<>EP@.YLB THEN IDENT:=FALSE;
52400              ZLB : IF E.ZLB<>EP@.ZLB THEN IDENT:=FALSE;
52500              XRO : IF E.XRO<>EP@.XRO THEN IDENT:=FALSE;
52600              YRO : IF E.YRO<>EP@.YRO THEN IDENT:=FALSE;
52700              ZRO : IF E.ZRO<>EP@.ZRO THEN IDENT:=FALSE;
52800              CHARS : IF E.CHARS<>EP@.CHARS THEN
52900                                              IDENT:=FALSE;
53000              TDET : IF E.DETECT<>EP@.RALINK@.DETECT THEN
53100                                              IDENT:=FALSE;
53200              TPEN : IF E.PEN<>EP@.RALINK@.PEN THEN IDENT:=FALSE;
53300              TCOL : IF E.COLOUR<>EP@.RALINK@.COLOUR THEN
53400                                              IDENT:=FALSE;
53500              ITALIC : IF E.ITALIC<>EP@.RALINK@.ITALIC THEN
53600                                              IDENT:=FALSE;
53700              CHARW : IF E.CHARW<>EP@.RALINK@.CHARW THEN
53800                                              IDENT:=FALSE;
53900              CHARH : IF E.CHARH<>EP@.RALINK@.CHARH THEN
54000                                              IDENT:=FALSE
54100              END
54200         END
54300         END
54400        END;
54500       OBJE :
54600        BEGIN
54700         IF (EP@.RALINK=NIL) AND
54800            (E.OEVAL*[ODET, OPEN, OCOL, OLST, OGREY]<>[])
54900         THEN IDENT:=FALSE ELSE
55000         BEGIN
55100         FOR ITOE:=OBJREF TO OSHAD DO
55200         IF IDENT THEN
55300         BEGIN IF ITOE IN E.OEVAL THEN
55400               CASE ITOE OF
55500               OBJREF : IF E.OBJREF<>EP@.OBJREF THEN
55600                                              IDENT:=FALSE;
55700               TMATRIX : IF E.TMATRIX<>EP@.TMATRIX THEN
55800                                              IDENT:=FALSE;
55900               ODET : IF E.DETECT<>EP@.RALINK@.DETECT THEN
56000                                              IDENT:=FALSE;
56100               OPEN : IF E.PEN<>EP@.RALINK@.PEN THEN IDENT:=FALSE;
56200               OCOL : IF E.COLOUR<>EP@.RALINK@.COLOUR THEN
56300                                              IDENT:=FALSE;
56400               OLST : IF E.OLST<>EP@.RALINK@.OLST THEN
56500                                              IDENT:=FALSE;
56600               OGREY : IF E.OGREY<>EP@.RALINK@.OGREY THEN
56700                                              IDENT:=FALSE
56800               END
56900         END
57000         END
57100        END
57200       END;
57300       IF IDENT THEN CMPIT:=TRUE
57400              ELSE CMPIT:=FALSE
57500       END ELSE CMPIT:=FALSE
57600 END;
      #
```

```
57800 FUNCTION MEMBERE (EP : EPTR;  VAR IDEP : EPTR;
57900                   OP :OPTR) : BOOLEAN;
58000  (* OBJECT OP@ IS SEARCHED FOR AN ELEMENT, THAT IS EQUAL     *)
58100  (* TO ELEMENT EP@.                                          *)
58200  (* IDEP BECOMES THE POINTER TO THAT ELEMENT IN OBJECT OP@   *)
58300 VAR   R : @REPR;
58400       IDENT : BOOLEAN;
58500       EPF, EPB : EPTR;
58600       CTRL, CTRP, ICTRP : COORDTRP;
58700 BEGIN CASE EP@.ELEMENTTYPE OF
58800       POINTE     : IDEP:=OP@.PLINK;
58900       LINEE      : IDEP:=OP@.LLINK;
59000       CONTOURE : IDEP:=OP@.CLINK;
59100       TEXTE      : IDEP:=OP@.TLINK;
59200       OBJE       : IDEP:=OP@.OELINK
59300       END;
59400       IDENT:=FALSE;
59500       EPF:=EP@.FLINK; EPB:=EP@.BLINK; R:=EP@.RALINK;
59600       IF EP@.ELEMENTTYPE=CONTOURE THEN
59700         BEGIN CTRP:=EP@.COORDTRL; CTRL:=CTRP END;
59800       WHILE (IDEP<>NIL) AND (NOT IDENT) DO
59900       BEGIN EP@.FLINK:=IDEP@.FLINK;
60000             EP@.BLINK:=IDEP@.BLINK;
60100             EP@.RALINK:=IDEP@.RALINK;
60200             IF EP@.ELEMENTTYPE=CONTOURE THEN
60300                EP@.COORDTRL:=IDEP@.COORDTRL;
60400             IF EP@=IDEP@ THEN
60500             BEGIN IF (R<>NIL) OR (IDEP@.RALINK<>NIL)
60600                   THEN
60700                        BEGIN
60800                         IF (R<>NIL) AND (IDEP@.RALINK<>NIL)
60900                         THEN
61000                          BEGIN IF R@=IDEP@.RALINK@
61100                                   THEN IDENT:=TRUE
61200                          END
61300                        END ELSE IDENT:=TRUE;
61400                   IF EP@.ELEMENTTYPE=CONTOURE THEN
61500                   BEGIN
61600                      ICTRP:=IDEP@.COORDTRL;
61700                      WHILE (CTRP<>NIL) AND IDENT DO
61800                       IF (CTRP@.X=ICTRP@.X) AND
61900                          (CTRP@.Y=ICTRP@.Y) AND
62000                          (CTRP@.Z=ICTRP@.Z) THEN
62100                       BEGIN CTRP:=CTRP@.FLINK;
62200                             ICTRP:=ICTRP@.FLINK
62300                       END ELSE IDENT:=FALSE
62400                   END
62500             END;
62600             IF NOT IDENT THEN IDEP:=IDEP@.FLINK
62700       END;
62800       EP@.FLINK:=EPF; EP@.BLINK:=EPB; EP@.RALINK:=R;
62900       IF EP@.ELEMENTTYPE=CONTOURE THEN EP@.COORDTRL:=CTRL;
63000       IF IDENT THEN MEMBERE:=TRUE
63100                ELSE MEMBERE:=FALSE
63200 END;
      #
```

B12

```
. . . REAL; VAR XT, YT, ZT, TT : REAL;
                     M : MATRIX);
 (* (XT, YT, ZT, TT):=(X, Y, Z, T)*M.                             *)
BEGIN XT:=X*M[1,1]+Y*M[2,1]+Z*M[3,1]+T*M[4,1];
      YT:=X*M[1,2]+Y*M[2,2]+Z*M[3,2]+T*M[4,2];
      ZT:=X*M[1,3]+Y*M[2,3]+Z*M[3,3]+T*M[4,3];
      TT:=X*M[1,4]+Y*M[2,4]+Z*M[3,4]+T*M[4,4]
END;

PROCEDURE MMULT (OM, M : MATRIX; VAR TM : MATRIX);
 (* TM:=OM*M.                                                     *)
VAR   I : INTEGER;
BEGIN FOR I:=1 TO 4 DO
            VMULT(OM[I,1], OM[I,2], OM[I,3], OM[I,4],
                  TM[I,1], TM[I,2], TM[I,3], TM[I,4], M)
END;

PROCEDURE TRAFE (EP, TEP : EPTR; TMATRIX : MATRIX);
 (* THE GRAPHICAL INFORMATION (COORD.S/TMATRIX) OF ELEMENT  *)
 (* TEP@ WILL BE DEFINED AS THE TRANSFORMED INFORMATION OF  *)
 (* ELEMENT EP@.                                            *)
VAR   TT : REAL;
      CTRP : COORDTRP;
BEGIN CASE EP@.ELEMENTTYPE OF
      POINTE    : VMULT(EP@.X, EP@.Y, EP@.Z, 1,TEP@.X, TEP@.Y,
                        TEP@.Z, TT, TMATRIX);
      LINEE     : BEGIN VMULT(EP@.XB, EP@.YB, EP@.ZB, 1,
                              TEP@.XB, TEP@.YB, TEP@.ZB, TT,
                              TMATRIX);
                        VMULT(EP@.XE, EP@.YE, EP@.ZE, 1,
                              TEP@.XE, TEP@.YE, TEP@.ZE, TT,
                              TMATRIX)
                  END;
      CONTOURE  : BEGIN CTRP:=TEP@.COORDTRL;
                        WHILE CTRP<>NIL DO
                        BEGIN
                         VMULT(CTRP@.X, CTRP@.Y, CTRP@.Z, 1,
                               CTRP@.X, CTRP@.Y, CTRP@.Z, TT,
                               TMATRIX);
                         CTRP:=CTRP@.FLINK
                        END
                  END;
      TEXTE     : BEGIN VMULT(EP@.XLO, EP@.YLO, EP@.ZLO, 1,
                              TEP@.XLO, TEP@.YLO, TEP@.ZLO,
                              TT, TMATRIX);
                        VMULT(EP@.XLB, EP@.YLB, EP@.ZLB, 1,
                              TEP@.XLB, TEP@.YLB, TEP@.ZLB,
                              TT, TMATRIX);
                        VMULT(EP@.XRO, EP@.YRO, EP@.ZRO, 1,
                              TEP@.XRO, TEP@.YRO, TEP@.ZRO,
                              TT, TMATRIX)
                  END;
      OBJE      : MMULT(EP@.TMATRIX, TMATRIX, TEP@.TMATRIX)
      END
END;
#
```

```
69000 PROCEDURE TRAFO (OP : OPTR; TMATRIX : MATRIX);
69100  (* A NEW OBJECT-DEFINITION WILL BE CREATED, THAT EXISTS    *)
69200  (* OF THE TRANSFORMED ELEMENTS OF OBJECT OP@ .             *)
69300  (* THE LOCATION OF THIS NEW OBJECT-DEFINITION WILL BE      *)
69400  (* WRITTEN IN THE OBJADM-RECORD OF OBJECT OBJ .            *)
69500 VAR   NEWO : OPTR; EP, TEP : EPTR;
69600       NEWA : @REPR;
69700 BEGIN
69800       NEW(NEWO);
69900       NEWO@.PLINK:=NIL;
70000       NEWO@.LLINK:=NIL;
70100       NEWO@.CLINK:=NIL;
70200       NEWO@.TLINK:=NIL;
70300       NEWO@.OELINK:=NIL;
70400       EP:=NIL;
70500       NEXTEL(EP, OP);
70600       WHILE EP<>NIL DO
70700       BEGIN CASE EP@.ELEMENTTYPE OF
70800             POINTE : BEGIN NEW(TEP, POINTE);
70900                           IF EP@.RALINK<>NIL THEN
71000                                NEW(NEWA, POINTE)
71100                      END;
71200             LINEE : BEGIN NEW(TEP, LINEE);
71300                           IF EP@.RALINK<>NIL THEN
71400                                NEW(NEWA, LINEE)
71500                     END;
71600             CONTOURE : BEGIN NEW(TEP, CONTOURE);
71700                              IF EP@.RALINK<>NIL THEN
71800                                   NEW(NEWA, CONTOURE)
71900                        END;
72000             TEXTE : BEGIN NEW(TEP, TEXTE);
72100                           IF EP@.RALINK<>NIL THEN
72200                                NEW(NEWA, TEXTE)
72300                     END;
72400             OBJE : BEGIN NEW(TEP, OBJE);
72500                          IF EP@.RALINK<>NIL THEN
72600                               NEW(NEWA, OBJE)
72700                    END
72800             END;
72900             TEP@:=EP@;
73000             IF EP@.RALINK<>NIL THEN
73100             BEGIN NEWA@:=EP@.RALINK@;
73200                   TEP@.RALINK:=NEWA
73300             END;
73400             IF EP@.ELEMENTTYPE=CONTOURE THEN
73500                COPYCTRS(TEP, EP@.COORDTRL);
73600             TRAFE(EP, TEP, TMATRIX);
73700             ADDE(TEP, NEWO);
73800             NEXTEL(EP, OP)
73900       END;
74000       ADMLOC(OBJ, NEWO)
74100 END;
      #
```

```
74300 FUNCTION MEMBER (VAR E : ELEMINF; OP : OPTR) : BOOLEAN;
74400  (* OBJECT OP@ WILL BE SEARCHED FOR AN ELEMENT, THAT IS      *)
74500  (* EQUAL TO THE ELEMENT-DESCRIPTION IN THE VARIABLE 'E'     *)
74600  (* E.LASTREF BECOMES THE POINTER TO THAT ELEMENT            *)
74700 VAR   EP : EPTR;
74800 BEGIN MEMBER:=FALSE;
74900       IF OP<>NIL THEN
75000       BEGIN
75100       IF E.LASTREF<>NIL THEN
75200       BEGIN EP:=E.LASTREF;
75300             IF (NOT CMPIT(E, EP)) OR (OBJ<>E.OBJ) THEN
75400             BEGIN E.LASTREF:=NIL;
75500                   IF MEMBER(E, OP) THEN MEMBER:=TRUE
75600             END ELSE MEMBER:=TRUE
75700       END ELSE
75800       BEGIN CASE E.ELEMENTTYPE OF
75900             POINTE     : EP:=OP@.PLINK;
76000             LINEE      : EP:=OP@.LLINK;
76100             CONTOURE : EP:=OP@.CLINK;
76200             TEXTE      : EP:=OP@.TLINK;
76300             OBJE       : EP:=OP@.OELINK
76400             END;
76500             WHILE (EP<>NIL) AND (NOT CMPIT(E, EP)) DO
76600                   EP:=EP@.FLINK;
76700             IF EP<>NIL THEN
76800             BEGIN MEMBER:=TRUE;
76900                   WRITELN(' EQUAL ELEMENT FOUND');
77000                   E.LASTREF:=EP
77100             END ELSE MEMBER:=FALSE
77200       END
77300       END
77400 END;
77500
77600 PROCEDURE DEL (VAR E : ELEMINF; OP : OPTR);
77700  (* IN OBJECT OP@   WILL BE SEARCHED WITH PROCEDURE          *)
77800  (* 'MEMBER' FOR AN ELEMENT, THAT IS, FOR THE ITEMS IN       *)
77900  (* E. VAL, IDENTICAL TO THE ELEMENT DESCRIBED IN THE        *)
78000  (* VARIABLE 'E'.                                           *)
78100  (* WHEN THE PROCEDURE 'MEMBER' IS ABLE TO DELIVER THE       *)
78200  (* LOCATION (E.LASTREF) OF AN ELEMENT, IT WILL BE           *)
78300  (* DELETED WITH PROCEDURE 'DELE'.                          *)
78400 BEGIN
78500       IF MEMBER(E, OP) THEN DELE(E.LASTREF, OP);
78600       E.LASTREF:=NIL
78700 END;
78800
78900 PROCEDURE ADD (VAR E : ELEMINF; OP : OPTR);
79000  (* IF E.LASTREF=NIL, THE ELEMENT-DESCRIPTION IN THE         *)
79100  (* VARIABLE 'E' WILL BE USED FOR CREATING A NEW ELEMENT.    *)
79200  (* IF E.LASTREF<>NIL, A NEW ELEMENT WILL BE CREATED BY      *)
79300  (* COPYING THE ELEMENT E.LASTREF@.                         *)
79400  (* IN BOTH CASES THE NEW ELEMENT WILL BE ADDED TO OBJECT    *)
79500  (* OP@   WITH PROCEDURE 'ADDE'.                            *)
79600 VAR   NEWE : EPTR;
79700       NEWA : @REPR;
79800       OBJREF : ONAME;
$
```

```
79900 BEGIN
80000       IF OP=NIL THEN
80100       BEGIN
80200             NEW(OP);
80300             OP@.PLINK:=NIL;
80400             OP@.LLINK:=NIL;
80500             OP@.CLINK:=NIL;
80600             OP@.TLINK:=NIL;
80700             OP@.OELINK:=NIL;
80800             ADMLOC(OBJ, OP)
80900       END;
81000       IF E.LASTREF=NIL THEN
81100       BEGIN
81200        CASE E.ELEMENTTYPE OF
81300        POINTE :
81400           BEGIN
81500             NEW(NEWE, POINTE);
81600             NEWE@.ELEMENTTYPE:=POINTE;
81700             NEWE@.X:=E.X;
81800             NEWE@.Y:=E.Y;
81900             NEWE@.Z:=E.Z;
82000            IF E.PVAL*[PDET, PPEN, PCOL]<>[] THEN
82100            BEGIN
82200             NEW(NEWA, POINTE);
82300             NEWA@.ELEMENTTYPE:=POINTE;
82400             IF PDET IN E.PVAL
82500               THEN NEWA@.DETECT:=E.DETECT
82600               ELSE NEWA@.DETECT:=DETDEF;
82700             IF PPEN IN E.PVAL
82800               THEN NEWA@.PEN:=E.PEN
82900               ELSE NEWA@.PEN:=PENDEF;
83000             IF PCOL IN E.PVAL
83100               THEN NEWA@.COLOUR:=E.COLOUR
83200               ELSE NEWA@.COLOUR:=COLDEF;
83300             NEWE@.RALINK:=NEWA;
83400            END ELSE NEWE@.RALINK:=NIL
83500           END;
83600        LINEE :
83700           BEGIN
83800            NEW(NEWE, LINEE);
83900            NEWE@.ELEMENTTYPE:=LINEE;
84000            NEWE@.XB:=E.XB; NEWE@.YB:=E.YB; NEWE@.ZB:=E.ZB;
84100            NEWE@.XE:=E.XE; NEWE@.YE:=E.YE; NEWE@.ZE:=E.ZE;
84200            IF E.LVAL*[LDET, LPEN, LCOL, LLST]<>[] THEN
84300            BEGIN
84400             NEW(NEWA, LINEE);
84500             NEWA@.ELEMENTTYPE:=LINEE;
84600             IF LDET IN E.LVAL THEN NEWA@.DETECT:=E.DETECT
84700                               ELSE NEWA@.DETECT:=DETDEF;
84800             IF LPEN IN E.LVAL THEN NEWA@.PEN:=E.PEN
84900                               ELSE NEWA@.PEN:=PENDEF;
85000             IF LCOL IN E.LVAL THEN NEWA@.COLOUR:=E.COLOUR
85100                               ELSE NEWA@.COLOUR:=COLDEF;
85200             IF LLST IN E.LVAL THEN NEWA@.LLST:=E.LLST
85300                               ELSE NEWA@.LLST.DASH1:=DASH1DEF;
85400             NEWE@.RALINK:=NEWA
85500            END ELSE NEWE@.RALINK:=NIL
85600           END;
            #
```

```
85700          CONTOURE :
85800             BEGIN
85900              NEW(NEWE, CONTOURE);
86000              NEWE@.ELEMENTTYPE:=CONTOURE;
86100              COPYCTRS(NEWE, E.COORDTRL);
86200              NEWE@.NOFCOORDTRS:=E.NOFCOORDTRS;
86300              IF E.CVAL*[CDET, CPEN, CCOL, CLST, CGREY]
86400                          <>[] THEN
86500              BEGIN
86600               NEW(NEWA, CONTOURE);
86700               NEWA@.ELEMENTTYPE:=CONTOURE;
86800               IF CDET IN E.CVAL THEN NEWA@.DETECT:=E.DETECT
86900                                 ELSE NEWA@.DETECT:=DETDEF;
87000               IF CPEN IN E.CVAL THEN NEWA@.PEN:=E.PEN
87100                                 ELSE NEWA@.PEN:=PENDEF;
87200               IF CCOL IN E.CVAL THEN NEWA@.COLOUR:=E.COLOUR
87300                                 ELSE NEWA@.COLOUR:=COLDEF;
87400               IF CLST IN E.CVAL THEN NEWA@.CLST:=E.CLST
87500                                 ELSE NEWA@.CLST.DASH1:=DASH1DEF;
87600               IF CGREY IN E.CVAL THEN NEWA@.CGREY:=E.CGREY
87700                                  ELSE NEWA@.CGREY:=GREYDEF;
87800               NEWE@.RALINK:=NEWA
87900              END ELSE NEWE@.RALINK:=NIL
88000             END;
88100          TEXTE :
88200             BEGIN
88300              NEW(NEWE, TEXTE);
88400              NEWE@.ELEMENTTYPE:=TEXTE;
88500              NEWE@.XLO:=E.XLO; NEWE@.YLO:=E.YLO; NEWE@.ZLO:=E.ZLO;
88600              NEWE@.XLB:=E.XLB; NEWE@.YLB:=E.YLB; NEWE@.ZLB:=E.ZLB;
88700              NEWE@.XRO:=E.XRO; NEWE@.YRO:=E.YRO; NEWE@.ZRO:=E.ZRO;
88800              NEWE@.CHARS:=E.CHARS;
88900              IF E.TVAL*[TDET, TPEN, TCOL, ITALIC, CHARW, CHARH]
89000                          <>[] THEN
89100              BEGIN
89200               NEW(NEWA, TEXTE);
89300               NEWA@.ELEMENTTYPE:=TEXTE;
89400               IF TDET IN E.TVAL THEN NEWA@.DETECT:=E.DETECT
89500                                 ELSE NEWA@.DETECT:=DETDEF;
89600               IF TPEN IN E.TVAL THEN NEWA@.PEN:=E.PEN
89700                                 ELSE NEWA@.PEN:=PENDEF;
89800               IF TCOL IN E.TVAL THEN NEWA@.COLOUR:=E.COLOUR
89900                                 ELSE NEWA@.COLOUR:=COLDEF;
90000               IF ITALIC IN E.TVAL THEN NEWA@.ITALIC:=E.ITALIC
90100                                 ELSE NEWA@.ITALIC:=ITALDEF;
90200               IF CHARW IN E.TVAL THEN NEWA@.CHARW:=E.CHARW
90300                                 ELSE NEWA@.CHARW:=CHARWDEF;
90400               IF CHARH IN E.TVAL THEN NEWA@.CHARH:=E.CHARH
90500                                 ELSE NEWA@.CHARH:=CHARHDEF;
90600               NEWE@.RALINK:=NEWA
90700              END ELSE NEWE@.RALINK:=NIL
90800             END;
          #
```

```
90900         OBJE :
91000            BEGIN
91100             OBJREF:=E.OBJREF;
91200             IF GETIND(OBJREF) THEN
91300             BEGIN
91400             NEW(NEWE, OBJE);
91500             NEWE@.ELEMENTTYPE:=OBJE;
91600             NEWE@.INDEX:=I;
91700             NEWE@.OBJREF:=E.OBJREF;
91800             NEWE@.TMATRIX:=E.TMATRIX;
91900             IF E.OEVAL*[ODET, OPEN, OCOL, OLST, OGREY]<>[] THEN
92000             BEGIN
92100              NEW(NEWA, OBJE);
92200              NEWA@.ELEMENTTYPE:=OBJE;
92300              IF ODET IN E.OEVAL THEN NEWA@.DETECT:=E.DETECT
92400                                 ELSE NEWA@.DETECT:=DETDEF;
92500              IF OPEN IN E.OEVAL THEN NEWA@.PEN:=E.PEN
92600                                 ELSE NEWA@.PEN:=PENDEF;
92700              IF OCOL IN E.OEVAL THEN NEWA@.COLOUR:=E.COLOUR
92800                                 ELSE NEWA@.COLOUR:=COLDEF;
92900              IF OLST IN E.OEVAL THEN NEWA@.OLST:=E.OLST
93000                                 ELSE NEWA@.OLST.DASH1:=DASH1DEF;
93100              IF OGREY IN E.OEVAL THEN NEWA@.OGREY:=E.OGREY
93200                                  ELSE NEWA@.OGREY:=GREYDEF;
93300              NEWE@.RALINK:=NEWA
93400             END ELSE NEWE@.RALINK:=NIL
93500             END (* ELSE REFERENCED OBJECT NOT DECLARED *)
93600            END
93700         END
93800        END ELSE
93900         (* IF E.LASTREF<>NIL THEN *)
94000        BEGIN
94100         CASE E.ELEMENTTYPE OF
94200         POINTE : NEW(NEWE, POINTE);
94300         LINEE : NEW(NEWE, LINEE);
94400         CONTOURE : NEW(NEWE, CONTOURE);
94500         TEXTE : NEW(NEWE, TEXTE);
94600         OBJE : NEW(NEWE, OBJE)
94700         END;
94800         NEWE@:=E.LASTREF@;
94900         IF NEWE@.RALINK<>NIL THEN
95000         BEGIN CASE NEWE@.ELEMENTTYPE OF
95100               POINTE : NEW(NEWA, POINTE);
95200               LINEE : NEW(NEWA, LINEE);
95300               CONTOURE : NEW(NEWA, CONTOURE);
95400               TEXTE : NEW(NEWA, TEXTE);
95500               OBJE : NEW(NEWA, OBJE)
95600               END;
95700               NEWA@:=E.LASTREF@.RALINK@;
95800              NEWE@.RALINK:=NEWA
95900         END;
96000         IF NEWE@.ELEMENTTYPE=CONTOURE THEN
96100             COPYCTRS(NEWE, E.LASTREF@.COORDTRL)
96200        END;
96300        ADDE(NEWE, OP);
96400        E.LASTREF:=NEWE;
96500        E.OBJ:=OBJ
96600 END;
      *
```

```
96800 PROCEDURE EMPTY (OP : OPTR);
96900  (* ALL ELEMENTS OF OBJECT OP@   WILL BE DELETED.          *)
97000  (* AN EMPTY OBJECT-DEFINITION WILL REMAIN.               *)
97100 VAR    EP : EPTR;
97200 BEGIN
97300       IF OP<>NIL THEN
97400       BEGIN
97500       RETSPACE(OP);
97600       OP@.PLINK:=NIL;
97700       OP@.LLINK:=NIL;
97800       OP@.CLINK:=NIL;
97900       OP@.TLINK:=NIL;
98000       OP@.OELINK:=NIL
98100       END (* ELSE NO OBJECT-DEFINITION PRESENT *)
98200 END;
98300
98400 PROCEDURE DIFF (OP1, OP2 : OPTR);
98500  (* A NEW OBJECT-DEFINITION WILL BE CREATED, THAT EXISTS    *)
98600  (* OF THOSE ELEMENTS OF OBJECT OP1@, WHO ARE NOT IN OBJECT *)
98700  (* OP2@.                                                  *)
98800  (* THE LOCATION OF THIS NEW OBJECT-DEFINITION WILL BE     *)
98900  (* WRITTEN IN THE OBJADM-RECORD OF OBJECT OBJ .           *)
99000 VAR   NEWO : OPTR;
99100       EP, IDEP : EPTR;
99200 BEGIN
99300       NEW(NEWO);
99400       NEWO@.PLINK:=NIL;
99500       NEWO@.LLINK:=NIL;
99600       NEWO@.CLINK:=NIL;
99700       NEWO@.TLINK:=NIL;
99800       NEWO@.OELINK:=NIL;
99900       COPYOBJ(OP1, NEWO);
100000       EP:=NIL; IDEP:=NIL;
100100       NEXTEL(EP, OP2);
100200       WHILE EP<>NIL DO
100300       BEGIN IF MEMBERE(EP, IDEP, NEWO) THEN DELE(IDEP, NEWO);
100400             NEXTEL(EP, OP2)
100500       END;
100600       OP:=NEWO;
100700       ADMLOC(OBJ, NEWO)
100800 END;
100900
101000 PROCEDURE UNION (OP1, OP2 : OPTR);
101100  (* A NEW OBJECT-DEFINITION WILL BE CREATED, THAT EXISTS    *)
101200  (* OF ALL ELEMENTS OF OBJECT OP1@ AND ALL ELEMENTS OF     *)
101300  (* OBJECT OP2@.                                           *)
101400  (* THE LOCATION OF THIS NEW OBJECT-DEFINITION WILL BE     *)
101500  (* WRITTEN IN THE OBJADM-RECORD OF OBJECT OBJ .           *)
101600 VAR   NEWO : OPTR;
101700 BEGIN
101800       NEW(NEWO);
101900       NEWO@.PLINK:=NIL;
102000       NEWO@.LLINK:=NIL;
102100       NEWO@.CLINK:=NIL;
102200       NEWO@.TLINK:=NIL;
102300       NEWO@.OELINK:=NIL;
102400       COPYOBJ(OP1, NEWO);
102500       COPYOBJ(OP2, NEWO);
102600       ADMLOC(OBJ, NEWO)
102700 END;
#
```

```
103000  (* BY CALLING TWICE PROCEDURE 'DIFF', AN OBJECT-DEFINITION *)
103100  (* WILL BE CREATED, THAT EXISTS OF THOSE ELEMENTS, THAT    *)
103200  (* ARE IN OBJECT OP1@ AS WELL AS IN OBJECT OP2@.           *)
103300  (* THE LOCATION OF THIS NEW OBJECT-DEFINITION WILL BE      *)
103400  (* WRITTEN IN THE OBJADM-RECORD OF OBJECT OBJ .            *)
103500 BEGIN DIFF(OP1, OP2);
103600       DIFF(OP1, OP)
103700 END;
103800
103900 PROCEDURE DELOBJ (OP : OPTR);
104000  (* THE OBJECT-DEFINITION OF OBJECT OP@  WILL BE DELETED   *)
104100  (* (IF NOFREF=0)                                          *)
104200 BEGIN
104300       IF OBJADM[I].NOFREF=0 THEN
104400            BEGIN RETSPACE(OBJADM[I].OBJLOC);
104500                  (* DISPOSE(OBJADM[I].OBJLOC) *)
104600                  OBJADM[I].OBJLOC:=NIL
104700            END;
104800            OBJADM[I].OBJ:='NO OBJECT '
104900 END;
105000
105100 FUNCTION DEFOBJ (OBJ : ONAME) : BOOLEAN;
105200  (* IN OBJADM WILL BE SEARCHED FOR AN UNUSED RECORD FOR     *)
105300  (* THE REGISTRATION OF OBJECT OBJ.                        *)
105400 VAR   I : INTEGER;
105500 BEGIN I:=1; DEFOBJ:=FALSE;
105600       WHILE I<>(NMAX+1) DO
105700       BEGIN IF (OBJADM[I].OBJ='NO OBJECT ') AND
105800                (OBJADM[I].OBJLOC=NIL) THEN
105900            BEGIN OBJADM[I].OBJ:=OBJ;
106000                  DEFOBJ:=TRUE;
106100                  I:=(NMAX+1)
106200            END ELSE I:=I+1
106300       END
106400 END;
106500
106600 PROCEDURE INIT;
106700  (* ALL RECORDS OF OBJADM WILL BE INITIALIZED.              *)
106800 VAR   I : INTEGER;
106900 BEGIN I:=1;
107000       WHILE I<>(NMAX+1) DO
107100       BEGIN OBJADM[I].OBJ:='NO OBJECT ';
107200            OBJADM[I].OBJLOC:=NIL;
107300            OBJADM[I].NOFREF:=0;
107400            I:=I+1
107500       END
107600 END;
107700
107800 PROCEDURE WRITEOBJ (OP : OPTR);
107900 VAR   EP : EPTR;
108000       CTRP : @COORDTR;
108100       I : INTEGER;
108200       OBJL : ONAME;
108300 BEGIN
108400        OBJL:=OBJ;
108500        WRITELN(' DESCRIPTION OF OBJECT : ',OBJL);
108600        EP:=NIL;
108700        NEXTEL(EP, OP);
108800        WHILE EP<>NIL DO
       #
```

```
108900        BEGIN
109000         CASE EP@.ELEMENTTYPE OF
109100         POINTE :
109200           BEGIN
109300            WRITELN(' POINT');
109400            WRITELN('   X=',EP@.X,'; Y=',EP@.Y,'; Z=',EP@.Z);
109500            IF EP@.RALINK=NIL THEN WRITELN('   NO ATTRIBUTES')
109600                              ELSE WRITELN('   WITH ATTRIBUTES')
109700           END;
109800         LINEE :
109900           BEGIN
110000            WRITELN(' LINE');
110100            WRITELN(' XB=',EP@.XB,' YB=',EP@.YB,' ZB=',EP@.ZB);
110200            WRITELN(' XE=',EP@.XE,' YE=',EP@.YE,' ZE=',EP@.ZE);
110300            IF EP@.RALINK=NIL THEN WRITELN('   NO ATTRIBUTES')
110400                              ELSE WRITELN('   WITH ATTRIBUTES')
110500           END;
110600         CONTOURE :
110700           BEGIN
110800            WRITELN(' CONTOUR WITH ',EP@.NOFCOORDTRS,' COORDTRS');
110900            CTRP:=EP@.COORDTRL;
111000            I:=1;
111100            WHILE CTRP<>NIL DO
111200            BEGIN
111300             WRITELN(I,'  X=',CTRP@.X,' Y=',CTRP@.Y,' Z=',CTRP@.Z);
111400             I:=I+1;
111500             CTRP:=CTRP@.FLINK
111600            END;
111700            IF EP@.RALINK=NIL THEN WRITELN('   NO ATTRIBUTES')
111800                              ELSE WRITELN('   WITH ATTRIBUTES')
111900           END;
112000         TEXTE :
112100           BEGIN
112200            WRITELN(' TEXT');
112300            WRITELN(' TEXT-AREA IS DEFINED BY : ');
112400            WRITELN(' XLO=',EP@.XLO,' YLO=',EP@.YLO,' ZLO=',
112500                                                EP@.ZLO);
112600            WRITELN(' XLB=',EP@.XLB,' YLB=',EP@.YLB,' ZLB=',
112700                                                EP@.ZLB);
112800            WRITELN(' XRO=',EP@.XRO,' YRO=',EP@.YRO,' ZRO=',
112900                                                EP@.ZRO);
113000            WRITELN(' TEXTSTRING=',EP@.CHARS);
113100            IF EP@.RALINK=NIL THEN WRITELN('   NO ATTRIBUTES')
113200                              ELSE WRITELN('   WITH ATTRIBUTES')
113300           END;
113400         OBJE :
113500           BEGIN
113600            WRITELN(' OBJECT-ELEMENT');
113700            OBJ:=EP@.OBJREF;
113800            WRITELN(' REFERENCE TO OBJECT :',OBJ);
113900            WRITELN(' TRANSFORMATION-MATRIX =');
114000            WRITELN(EP@.TMATRIX[1,1],EP@.TMATRIX[1,2],
114100                    EP@.TMATRIX[1,3],EP@.TMATRIX[1,4]);
114200            WRITELN(EP@.TMATRIX[2,1],EP@.TMATRIX[2,2],
114300                    EP@.TMATRIX[2,3],EP@.TMATRIX[2,4]);
114400            WRITELN(EP@.TMATRIX[3,1],EP@.TMATRIX[3,2],
114500                    EP@.TMATRIX[3,3],EP@.TMATRIX[3,4]);
114600            WRITELN(EP@.TMATRIX[4,1],EP@.TMATRIX[4,2],
114700                    EP@.TMATRIX[4,3],EP@.TMATRIX[4,4]);
114800            IF EP@.RALINK=NIL THEN WRITELN('   NO ATTRIBUTES')
114900                              ELSE WRITELN('   WITH ATTRIBUTES');
           #
```

```
115000            WRITEOBJ(OBJADM[EP@.INDEX].OBJLOC)
115100           END
115200         END;
115300         NEXTEL(EP, OP)
115400        END;
115500        WRITELN(' END OF ',OBJL); OBJ:=OBJL
115600 END;
115700
115800  (* DECLARATION OF GINO-F SUBROUTINES                        *)
115900
116000 PROCEDURE DISPLAY(OP : OPTR; TRM : MATRIX);
116100 VAR    EP : EPTR;
116200       M : MATRIX;
116300       X, Y, Z, TT, FX, FY, FZ : REAL;
116400       CTRP : COORDTRP;
116500 BEGIN EP:=NIL;
116600       NEXTEL(EP, OP);
116700       WHILE EP<>NIL DO
116800       BEGIN
116900        CASE EP@.ELEMENTTYPE OF
117000        POINTE : BEGIN
117100          VMULT(EP@.X, EP@.Y, EP@.Z, 1, X, Y, Z, TT, TRM);
117200          MOVTO3(X, Y, Z);
117300          DOT(1)
117400          END; (*P*)
117500        LINEE : BEGIN
117600          VMULT(EP@.XB,EP@.YB,EP@.ZB,1,X,Y,Z,TT,TRM);
117700          MOVTO3(X, Y, Z);
117800          VMULT(EP@.XE,EP@.YE,EP@.ZE,1,X,Y,Z,TT,TRM);
117900          LINTO3(X, Y, Z)
118000          END; (*L*)
118100        CONTOURE : BEGIN
118200          CTRP:=EP@.COORDTRL;
118300          VMULT(CTRP@.X,CTRP@.Y,CTRP@.Z,1,FX,FY,FZ,TT,TRM);
118400          CTRP:=CTRP@.FLINK;
118500          MOVTO3(FX, FY, FZ);
118600          WHILE CTRP<>NIL DO
118700          BEGIN
118800           VMULT(CTRP@.X,CTRP@.Y,CTRP@.Z,1,X,Y,Z,TT,TRM);
118900           LINTO3(X, Y, Z);
119000           CTRP:=CTRP@.FLINK
119100          END;
119200          LINTO3(FX, FY, FZ)
119300          END; (*C*)
119400        TEXTE :
119500         BEGIN
119600          VMULT(EP@.XLO,EP@.YLO,EP@.ZLO,1,X,Y,Z,TT,TRM);
119700          MOVTO(X, Y, Z);
119800          CHAARR(EP@.CHARS, 10, 1)
119900         END; (*T*)
120000        OBJE :
120100         BEGIN
120200          MMULT(EP@.TMATRIX, TRM, M);
120300           (* M:=EP@.TMATRIX*TRM *)
120400          DISPLAY(OBJADM[EP@.INDEX].OBJLOC, M)
120500         END (*O*)
120600        END; (*CASE*)
120700        NEXTEL(EP, OP)
120800       END (*WHILE*)
120900 END; (*DISPLAY*)
       *
```

```
121100 PROCEDURE CREATEMATRIX(VAR M : MATRIX);
121200 VAR    ENDM : BOOLEAN;
121300        T : MATRIX;
121400        RX, RY, RZ : REAL;
121500 BEGIN M[1,1]:=1; M[1,2]:=0; M[1,3]:=0; M[1,4]:=0;
121600        M[2,1]:=0; M[2,2]:=1; M[2,3]:=0; M[2,4]:=0;
121700        M[3,1]:=0; M[3,2]:=0; M[3,3]:=1; M[3,4]:=0;
121800        M[4,1]:=0; M[4,2]:=0; M[4,3]:=0; M[4,4]:=1;
121900        CH:=' ';
122000        WRITELN(' OBJECT-TRANSFORMATION - YES OR NO?');
122100        WHILE CH=' ' DO READ(CH);
122200        WHILE NOT EOLN(INPUT) DO READ(ICH);
122300        IF CH='Y' THEN
122400        BEGIN
122500         ENDM:=FALSE;
122600         WRITELN(' BEGIN OF TMATRIX-DEFINITION');
122700         WHILE NOT ENDM DO
122800         BEGIN
122900          CH:=' ';
123000          WRITELN(' ENTER: ROT, TRANSL, SCALE OR ENDM');
123100          WHILE CH=' ' DO READ(CH);
123200          T[1,4]:=0; T[2,4]:=0; T[3,4]:=0; T[4,4]:=1;
123300          WHILE NOT EOLN(INPUT) DO READ(ICH);
123400          CASE CH OF
123500          'R' : BEGIN
123600                 T[4,1]:=0; T[4,2]:=0; T[4,3]:=0;
123700                 CH:=' ';
123800                 WRITELN(' WHICH AXIS : X, Y, Z');
123900                 WHILE CH=' ' DO READ(CH);
124000                 WHILE NOT EOLN(INPUT) DO READ(ICH);
124100                 CASE CH OF
124200                 'X' : BEGIN
124300                        WRITELN(' HOW MANY DEGREES');
124400                        READ(RX);
124500                        RX:=RX*PI/180;
124600                        T[1,1]:=1; T[1,2]:=0; T[1,3]:=0;
124700                        T[2,1]:=0; T[2,2]:=COS(RX); T[2,3]:=-SIN(RX);
124800                        T[3,1]:=0; T[3,2]:=-T[2,3]; T[3,3]:=T[2,2]
124900                       END;
125000                 'Y' : BEGIN
125100                        WRITELN(' HOW MANY DEGREES');
125200                        READ(RY);
125300                        RY:=RY*PI/180;
125400                        T[1,1]:=COS(RY); T[1,2]:=0; T[1,3]:=SIN(RY);
125500                        T[2,1]:=0; T[2,2]:=1; T[2,3]:=0;
125600                        T[3,1]:=-T[1,3]; T[3,2]:=0; T[3,3]:=T[1,1]
125700                       END;
125800                 'Z' : BEGIN
125900                        WRITELN(' HOW MANY DEGREES');
126000                        READ(RZ);
126100                        RZ:=RZ*PI/180;
126200                        T[1,1]:=COS(RZ); T[1,2]:=-SIN(RZ); T[1,3]:=0;
126300                        T[2,1]:=-T[1,2]; T[2,2]:=T[1,1]; T[2,3]:=0;
126400                        T[3,1]:=0; T[3,2]:=0; T[3,3]:=1
126500                       END
126600                 END; (*CASE*)
126700                 MMULT(M, T, M)
126800                END; (*R*)
       #
```

```
126900          'T' : BEGIN
127000                T[1,1]:=1; T[1,2]:=0; T[1,3]:=0;
127100                T[2,1]:=0; T[2,2]:=1; T[2,3]:=0;
127200                T[3,1]:=0; T[3,2]:=0; T[3,3]:=1;
127300                WRITELN(' ENTER TX-VALUE');
127400                READ(T[4,1]);
127500                WRITELN(' ENTER TY-VALUE');
127600                READ(T[4,2]);
127700                WRITELN(' ENTER TZ-VALUE');
127800                READ(T[4,3]);
127900                MMULT(M, T, M)
128000              END; (*T*)
128100          'S' : BEGIN
128200                T[1,2]:=0; T[1,3]:=0;
128300                T[2,1]:=0; T[2,3]:=0;
128400                T[3,1]:=0; T[3,2]:=0;
128500                T[4,1]:=0; T[4,2]:=0; T[4,3]:=0;
128600                WRITELN(' ENTER SX-VALUE');
128700                READ(T[1,1]);
128800                WRITELN(' ENTER SY-VALUE');
128900                READ(T[2,2]);
129000                WRITELN(' ENTER SZ-VALUE');
129100                READ(T[3,3]);
129200                MMULT(M, T, M)
129300              END; (*S*)
129400          'E' : ENDM:=TRUE
129500         END (*CASE*)
129600       END; (*WHILE*)
129700       WRITELN(' END OF TMATRIX-DEFINITION')
129800     END (*IF*)
129900 END; (*CREATEMATRIX*)
130000
130100 PROCEDURE READPOINT(VAR E : ELEMINF; ADDEL : BOOLEAN);
130200 VAR   ENDE : BOOLEAN;
130300 BEGIN E.LASTREF:=NIL;
130400       E.OBJ:='NO OBJECT ';
130500       E.ELEMENTTYPE:=POINTE;
130600          WRITELN(' BEGIN OF POINT-DESCRIPTION');
130700          E.PVAL:=[];
130800          ENDE:=FALSE;
130900          WHILE NOT ENDE DO
131000          BEGIN
131100           CH:=' '; I:=1; ITEM:=0;
131200           WRITELN(' ENTER POINT-ITEM');
131300           WHILE CH=' ' DO READ(CH);
131400           WHILE NOT EOLN(INPUT) AND (CH<>'=') DO
131500           BEGIN IF I<4 THEN BEGIN ITEM:=ITEM+ORD(CH); I:=I+1
131600                             END;
131700                 READ(CH)
131800           END;
131900           CASE ITEM OF
132000           231   : BEGIN (*X*)
132100                    E.PVAL:=E.PVAL+[X];
132200                    IF CH='=' THEN READ(E.X)
132300                   END;
132400           232   : BEGIN (*Y*)
132500                    E.PVAL:=E.PVAL+[Y];
132600                    IF CH='=' THEN READ(E.Y);
132700                   END;
#
```

```
132800            233   : BEGIN (*Z*)
132900                     E.PVAL:=E.PVAL+[Z];
133000                     IF CH='=' THEN READ(E.Z);
133100                    END;
133200            625   : BEGIN (*PEN*)
133300                     E.PVAL:=E.PVAL+[PPEN];
133400                     IF CH='=' THEN READ(E.PEN);
133500                    END;
133600            620   : BEGIN (*COL*)
133700                     E.PVAL:=E.PVAL+[PCOL];
133800                     IF CH='=' THEN READ(E.COLOUR);
133900                    END;
134000            606   : BEGIN (*END*)
134100                     WRITELN(' END OF POINT-DESCRIPTION');
134200                     ENDE:=TRUE
134300                    END
134400           END (*CASE*)
134500          END; (*WHILE*)
134600          IF ADDEL THEN
134700          BEGIN IF E.PVAL*[X, Y, Z]<>[X, Y, Z] THEN
134800                BEGIN
134900                 WRITELN(' POINT-DESCRIPTION NOT COMPLETE');
135000                 WRITELN(' TRY AGAIN');
135100                 READPOINT(E, TRUE)
135200                END
135300          END
135400 END; (*READPOINT*)
135500
135600 PROCEDURE READLINE(VAR E : ELEMINF; ADDEL : BOOLEAN);
135700 VAR   ENDE : BOOLEAN;
135800 BEGIN E.LASTREF:=NIL;
135900       E.OBJ:='NO OBJECT ';
136000       E.ELEMENTTYPE:=LINEE;
136100          WRITELN(' BEGIN OF LINE-DESCRIPTION');
136200          E.LVAL:=[];
136300          ENDE:=FALSE;
136400          WHILE NOT ENDE DO
136500          BEGIN
136600           CH:=' '; I:=1; ITEM:=0;
136700           WRITELN(' ENTER LINE-ITEM');
136800           WHILE CH=' ' DO READ(CH);
136900           WHILE NOT EOLN(INPUT) AND (CH<>'=') DO
137000           BEGIN IF I<4 THEN BEGIN ITEM:=ITEM+ORD(CH); I:=I+1
137100                               END;
137200                 READ(CH)
137300           END;
137400           CASE ITEM OF
137500           425   : BEGIN (*XB*)
137600                    E.LVAL:=E.LVAL+[XB];
137700                    IF CH='=' THEN READ(E.XB)
137800                   END;
137900           426   : BEGIN (*YB*)
138000                    E.LVAL:=E.LVAL+[YB];
138100                    IF CH='=' THEN READ(E.YB)
138200                   END;
138300           427   : BEGIN (*ZB*)
138400                    E.LVAL:=E.LVAL+[ZB];
138500                    IF CH='=' THEN READ(E.ZB)
138600                   END;
          #
```

```
138700            428   : BEGIN (*XE*)
138800                      E.LVAL:=E.LVAL+[XE];
138900                      IF CH='=' THEN READ(E.XE)
139000                    END;
139100            429   : BEGIN (*YE*)
139200                      E.LVAL:=E.LVAL+[YE];
139300                      IF CH='=' THEN READ(E.YE)
139400                    END;
139500            430   : BEGIN (*ZE*)
139600                          E.LVAL:=E.LVAL+[ZE];
139700                          IF CH='=' THEN READ(E.ZE)
139800                    END;
139900            625   : BEGIN (*PEN*)
140000                          E.LVAL:=E.LVAL+[LPEN];
140100                          IF CH='=' THEN READ(E.PEN)
140200                    END;
140300            620   : BEGIN (*COL*)
140400                          E.LVAL:=E.LVAL+[LCOL];
140500                          IF CH='=' THEN READ(E.COLOUR)
140600                    END;
140700            664   : BEGIN (*LST*)
140800                          WRITELN(' ENTER PERIOD');
140900                          READ(LST@.PERIOD);
141000                          WRITELN(' ENTER DASH1: 0..100');
141100                          READ(LST@.DASH1);
141200                          WRITELN(' ENTER GAP: 0..(100-DASH1)');
141300                          READ(LST@.GAP);
141400                    WRITELN(' ENTER DASH2: 0..(100-DASH1-GAP)');
141500                          READ(LST@.DASH2);
141600                          E.LLST:=LST;
141700                          E.LVAL:=E.LVAL+[LLST]
141800                    END;
141900            606   : BEGIN (*END*)
142000                          WRITELN(' END OF LINE-DESCRIPTION');
142100                          ENDE:=TRUE
142200                    END
142300           END (*CASE*)
142400         END; (*WHILE*)
142500         (* CONTROLEER OF BEGINPUNT<>EINDPUNT *)
142600         IF ADDEL THEN
142700         BEGIN IF E.LVAL*[XB,YB,ZB,XE,YE,ZE]<>
142800                  [XB,YB,ZB,XE,YE,ZE] THEN
142900                BEGIN
143000                 WRITELN(' LINE-DESCRIPTION NOT COMPLETE');
143100                 WRITELN(' TRY AGAIN');
143200                 READLINE(E, TRUE)
143300                END
143400         END
143500 END; (*LINE*)
143600
143700 PROCEDURE READCONTOUR(VAR E : ELEMINF; ADDEL : BOOLEAN);
143800 VAR   ENDE : BOOLEAN;
143900       NEWCTR1, NEWCTR2 : COORDTRP;
144000       N : INTEGER;
144100 BEGIN E.LASTREF:=NIL;
144200       E.OBJ:='NO OBJECT ';
144300       E.ELEMENTTYPE:=CONTOURE;
144400          WRITELN(' BEGIN OF CONTOUR-DESCRIPTION');
144500          E.CVAL:=[];
144600          ENDE:=FALSE;
#
```

```
144700          WHILE NOT ENDE DO
144800          BEGIN
144900           CH:=' '; I:=1; ITEM:=0;
145000           WRITELN(' ENTER CONTOUR-ITEM');
145100           WHILE CH=' ' DO READ(CH);
145200           WHILE NOT EOLN(INPUT) AND (CH<>'=') DO
145300           BEGIN IF I<4 THEN BEGIN ITEM:=ITEM+ORD(CH); I:=I+1
145400                         END;
145500                 READ(CH)
145600           END;
145700           CASE ITEM OF
145800           623   : BEGIN (*COO*)
145900                    WRITELN(' HOW MANY COORDINATE-TRIPLES');
146000                    READ(E.NOFCOORDTRS);
146100                    IF E.NOFCOORDTRS<3 THEN
146200                    WRITELN(' YOU HAVE TO ENTER AT LEAST 3 POINTS')
146300                    ELSE
146400                    BEGIN
146500                     NEW(NEWCTR1);
146600                     E.COORDTRL:=NEWCTR1;
146700                     N:=1;
146800                     WHILE N<=E.NOFCOORDTRS DO
146900                     BEGIN
147000                      WRITELN(' ENTER X-VALUE FOR POINT NO.',N);
147100                      READ(NEWCTR1@.X);
147200                      WRITELN(' ENTER Y-VALUE FOR POINT NO.',N);
147300                      READ(NEWCTR1@.Y);
147400                      WRITELN(' ENTER Z-VALUE FOR POINT NO.',N);
147500                      READ(NEWCTR1@.Z);
147600                      IF N=E.NOFCOORDTRS THEN NEWCTR1@.FLINK:=NIL
147700                      ELSE BEGIN NEWCTR2:=NEWCTR1;
147800                                 NEW(NEWCTR1);
147900                                 NEWCTR2@.FLINK:=NEWCTR1
148000                           END;
148100                      N:=N+1
148200                     END; (*WHILE*)
148300                     E.CVAL:=E.CVAL+[COORDTRS];
148400                     WRITELN(' INPUT OF POINTS COMPLETED')
148500                    END (*IF*)
148600                   END;
148700           625   : BEGIN (*PEN*)
148800                    IF CH='=' THEN READ(E.PEN);
148900                    E.CVAL:=E.CVAL+[CPEN]
149000                   END;
149100           620   : BEGIN (*COL*)
149200                    IF CH='=' THEN READ(E.COLOUR);
149300                    E.CVAL:=E.CVAL+[CCOL]
149400                   END;
149500           664   : BEGIN (*LST*)
149600                         WRITELN(' ENTER PERIOD');
149700                         READ(LST@.PERIOD);
149800                         WRITELN(' ENTER DASH1: 0..100');
149900                         READ(LST@.DASH1);
150000                         WRITELN(' ENTER GAP: 0..(100-DASH1)');
150100                         READ(LST@.GAP);
150200                   WRITELN(' ENTER DASH2: 0..(100-DASH1-GAP)');
150300                         READ(LST@.DASH2);
150400                         E.CLST:=LST;
150500                         E.CVAL:=E.CVAL+[CLST]
150600                   END;
          $
```

```
150700           613   : BEGIN (*GRE*)
150800                    IF CH='=' THEN READ(E.CGREY);
150900                    E.CVAL:=E.CVAL+[CGREY]
151000                   END;
151100           606   : BEGIN (*END*)
151200                         WRITELN(' END OF CONTOUR-DESCRIPTION');
151300                         ENDE:=TRUE
151400                   END
151500          END (*CASE*)
151600         END; (*WHILE*)
151700         (* CONTROLEER OF AAN DE DEFINITIE VAN EEN CONTOUR-
151800            ELEMENT WORDT VOLDAAN :
151900               - ALLE HOEKPUNTEN IN EEN VLAK;
152000               - OMTREKLIJNSTUKKEN MOGEN ELKAAR NIET SNIJDEN *)
152100         IF ADDEL THEN
152200         BEGIN IF E.CVAL*[COORDTRS]<>[COORDTRS] THEN
152300               BEGIN
152400                WRITELN(' CONTOUR-DESCRIPTION NOT COMPLETE');
152500                WRITELN(' TRY AGAIN');
152600                READCONTOUR(E, TRUE)
152700               END
152800         END
152900 END; (*CONTOUR*)
153000
153100 PROCEDURE READTEXT(VAR E : ELEMINF; ADDEL : BOOLEAN);
153200 VAR   ENDE : BOOLEAN;
153300 BEGIN E.LASTREF:=NIL;
153400       E.OBJ:='NO OBJECT ';
153500       E.ELEMENTTYPE:=TEXTE;
153600          WRITELN(' BEGIN OF TEXT-DESCRIPTION');
153700          E.TVAL:=[];
153800          ENDE:=FALSE;
153900          WHILE NOT ENDE DO
154000          BEGIN
154100           CH:=' '; I:=1; ITEM:=0;
154200           WRITELN(' ENTER TEXT-ITEM');
154300           WHILE CH=' ' DO READ(CH);
154400           WHILE NOT EOLN(INPUT) AND (CH<>'=') DO
154500           BEGIN IF I<4 THEN
154600                 BEGIN ITEM:=ITEM+ORD(CH);
154700                       I:=I+1
154800                 END;
154900                 READ(CH)
155000           END;
155100           CASE ITEM OF
155200           655   : BEGIN (*POS*)
155300                      WRITELN(' ENTER XLO-VALUE');
155400                      READ(E.XLO);
155500                      WRITELN(' ENTER YLO-VALUE');
155600                      READ(E.YLO);
155700                      WRITELN(' ENTER ZLO-VALUE');
155800                      READ(E.ZLO);
155900                      E.TVAL:=E.TVAL+[XLO,YLO,ZLO];
156000                      CH:=' ';
156100                      WRITELN(' FURTHER DEF. OF TEXT-AREA - YES/NO?');
156200                      WHILE CH=' ' DO READ(CH);
156300                      WHILE NOT EOLN(INPUT) DO READ(ICH);
#
```

```
156400                    IF CH='Y' THEN
156500                    BEGIN
156600                     WRITELN(' ENTER XRO-VALUE(WRITE-DIRECTION)');
156700                     READ(E.XRO);
156800                     WRITELN(' ENTER YRO-VALUE');
156900                     READ(E.YRO);
157000                     WRITELN(' ENTER ZRO-VALUE');
157100                     READ(E.ZRO);
157200                     WRITELN(' ENTER XLB-VALUE(VERT. DIRECTION)');
157300                     READ(E.XLB);
157400                     WRITELN(' ENTER YLB-VALUE');
157500                     READ(E.YLB);
157600                     WRITELN(' ENTER ZLB-VALUE');
157700                     READ(E.ZLB);
157800                     E.TVAL:=E.TVAL+[XRO,YRO,ZRO,XLB,YLB,ZLB]
157900                    END ELSE
158000                    BEGIN
158100                     E.XRO:=0; E.YRO:=0; E.ZRO:=0;
158200                     E.XLB:=0; E.YLB:=0; E.ZLB:=0
158300                    END
158400                   END;
158500            670   : BEGIN (*STR*)
158600                     WRITELN(' ENTER TEXTSTRING');
158700                     CH:=' '; I:=1; E.CHARS:='          ';
158800                     WHILE CH=' ' DO READ(CH);
158900                     WHILE NOT EOLN(INPUT) DO
159000                     BEGIN IF I<11 THEN
159100                           BEGIN E.CHARS[I]:=CH; I:=I+1 END;
159200                           READ(CH)
159300                     END;
159400                     E.TVAL:=E.TVAL+[CHARS]
159500                   END;
159600            625   : BEGIN (*PEN*)
159700                     IF CH='=' THEN READ(E.PEN);
159800                     E.TVAL:=E.TVAL+[TPEN]
159900                   END;
160000            620   : BEGIN (*COL*)
160100                     IF CH='=' THEN READ(E.COLOUR);
160200                     E.TVAL:=E.TVAL+[TCOL]
160300                   END;
160400            621   : BEGIN (*ITA*)
160500                           IF CH='=' THEN READ(E.ITALIC);
160600                           E.TVAL:=E.TVAL+[ITALIC]
160700                   END;
160800            626   : BEGIN (*CWI*)
160900                           IF CH='=' THEN READ(E.CHARW);
161000                           E.TVAL:=E.TVAL+[CHARW]
161100                   END;
161200            599   : BEGIN (*CHE*)
161300                           IF CH='=' THEN READ(E.CHARH);
161400                           E.TVAL:=E.TVAL+[CHARH]
161500                   END;
161600            606   : BEGIN (*END*)
161700                     WRITELN(' END OF TEXT-DESCRIPTION');
161800                     ENDE:=TRUE
161900                   END
162000           END (*CASE*)
162100         END; (*WHILE*)
         *
```

```
162200          (* CONTROLEER OF AAN DE DEFINITIE VAN EEN TEXT-
162300             ELEMENT WORDT VOLDAAN :
162400                  - PLO, PLB, PRO NIET OP EEN LIJN
162500                  - PLO, PLB, PRO NIET IDENTIEK                *)
162600          IF ADDEL THEN
162700          BEGIN IF E.TVAL*[XLO,YLO,ZLO,CHARS]<>
162800                          [XLO,YLO,ZLO,CHARS] THEN
162900                BEGIN WRITELN(' TEXT-DESCRIPTION NOT COMPLETE');
163000                      WRITELN(' TRY AGAIN');
163100                      READTEXT(E, TRUE)
163200                END
163300          END
163400 END; (*TEXT*)
163500
163600 PROCEDURE READOBJEL(VAR E : ELEMINF; ADDEL : BOOLEAN);
163700 VAR   ENDE : BOOLEAN;
163800 BEGIN E.LASTREF:=NIL;
163900       E.OBJ:='NO OBJECT ';
164000       E.ELEMENTTYPE:=OBJE;
164100          WRITELN(' BEGIN OF OBJECT-ELEMENT DESCRIPTION');
164200          E.OEVAL:=[];
164300          ENDE:=FALSE;
164400          WHILE NOT ENDE DO
164500          BEGIN
164600           CH:=' '; I:=1; ITEM:=0;
164700           WRITELN(' ENTER OBJECT-ELEMENT ITEM');
164800           WHILE CH=' ' DO READ(CH);
164900           WHILE NOT EOLN(INPUT) AND (CH<>'=') DO
165000           BEGIN IF I<4 THEN BEGIN ITEM:=ITEM+ORD(CH); I:=I+1
165100                             END;
165200                 READ(CH)
165300           END;
165400           CASE ITEM OF
165500           617   : BEGIN (*OBJ*)
165600                    WRITELN(' ENTER NAME OF REF. OBJECT');
165700                    CH:=' '; I:=1; E.OBJREF:='          ';
165800                    WHILE CH=' ' DO READ(CH);
165900                    WHILE NOT EOLN(INPUT) DO
166000                    BEGIN IF I<11 THEN
166100                          BEGIN E.OBJREF[I]:=CH; I:=I+1 END;
166200                          READ(CH)
166300                    END;
166400                    E.OEVAL:=E.OEVAL+[OBJREF]
166500                   END;
166600           632   : BEGIN (*TMA*)
166700                    CREATEMATRIX(TM);
166800                    E.TMATRIX:=TM;
166900                    E.OEVAL:=E.OEVAL+[TMATRIX]
167000                   END;
167100           625   : BEGIN (*PEN*)
167200                         IF CH='=' THEN READ(E.PEN);
167300                         E.OEVAL:=E.OEVAL+[OPEN]
167400                   END;
167500           620   : BEGIN (*COL*)
167600                         IF CH='=' THEN READ(E.COLOUR);
167700                         E.OEVAL:=E.OEVAL+[OCOL]
167800                   END;
      #
```

```
167900           664   : BEGIN (*LST*)
168000                         WRITELN(' ENTER PERIOD');
168100                         READ(LST@.PERIOD);
168200                         WRITELN(' ENTER DASH1: 0..100');
168300                         READ(LST@.DASH1);
168400                         WRITELN(' ENTER GAP: 0..(100-DASH1)');
168500                         READ(LST@.GAP);
168600                   WRITELN(' ENTER DASH2: 0..(100-DASH1-GAP)');
168700                         READ(LST@.DASH2);
168800                         E.OLST:=LST;
168900                         E.OEVAL:=E.OEVAL+[OLST]
169000                   END;
169100           613   : BEGIN (*GRE*)
169200                         IF CH='=' THEN READ(E.OGREY);
169300                         E.OEVAL:=E.OEVAL+[OGREY]
169400                   END;
169500           606   : BEGIN (*END*)
169600                         WRITELN(' END OF OBJECT-EL. DESCRIPTION');
169700                         ENDE:=TRUE
169800                   END
169900          END (*CASE*)
170000         END; (*WHILE*)
170100         (* CONTROLEER OP DE EVENTUELE AANWEZIGHEID VAN EEN
170200            LOOP.                                          *)
170300         IF ADDEL THEN
170400         BEGIN IF E.OEVAL*[OBJREF,TMATRIX]<>
170500                           [OBJREF,TMATRIX] THEN
170600               BEGIN
170700                WRITELN(' OBJECT-EL. DESCRIPTION NOT COMPLETE');
170800                WRITELN(' TRY AGAIN');
170900                READOBJEL(E, TRUE)
171000               END
171100         END
171200 END; (*OBJEL*)
171300
171400 FUNCTION READOBJ(VAR OP : OPTR) : BOOLEAN;
171500 BEGIN CH:=' '; I:=1; OBJ:='          ';
171600       WRITELN(' ENTER OBJECTNAME');
171700       WHILE CH=' ' DO READ(CH);
171800       WHILE NOT EOLN(INPUT) DO
171900       BEGIN IF I<11 THEN BEGIN OBJ[I]:=CH; I:=I+1 END;
172000             READ(CH)
172100       END;
172200       WRITELN(' OBJECTNAME : ',OBJ);
172300       IF LOCOBJ(OBJ, OP) THEN READOBJ:=TRUE
172400                          ELSE READOBJ:=FALSE
172500 END; (* READOBJ *)
       *
```

```
172700 BEGIN
172800       WRITELN(' ENTER CHAR TO OPEN FILE INPUT');
172900       READ(CH);
173000       READ(CH); (* OPEN FILE *)
173100       ENDOFPR:=FALSE;
173200       INIT;
173300       WHILE NOT ENDOFPR DO
173400       BEGIN
173500         WRITELN(' NEXT COMMAND PLEASE');
173600        CH:=' '; I:=1; OPERATION:=0;
173700        WHILE CH=' ' DO READ(CH);
173800        WHILE NOT EOLN(INPUT) DO
173900        BEGIN IF I<4 THEN BEGIN OPERATION:=OPERATION+ORD(CH);
174000                              I:=I+1
174100                         END;
174200             READ(CH)
174300        END;
174400        IF I<4 THEN WRITELN(' INVALID COMMAND');
174500        CASE OPERATION OF
174600        615 : (*INI*)
174700                INIT;
174800        640 : (*NEW*)
174900           BEGIN IF READOBJ(OP) THEN
175000                 WRITELN(' OBJECT',OBJ,'ALREADY DECLARED')
175100                 ELSE
175200                 BEGIN
175300                IF DEFOBJ(OBJ) THEN
175400                WRITELN(' OBJECT ',OBJ,' IN OBJADM')
175500                ELSE WRITELN(' NO OBJADM POSSIBLE')
175600                 END
175700           END;
175800        626 : (*REM*)
175900          BEGIN IF READOBJ(OP) THEN
176000                BEGIN
176100                DELOBJ(OP);
176200                WRITELN(' OBJECT ',OBJ,' REMOVED')
176300                END ELSE WRITELN(' OBJECT',OBJ,'NOT DECLARED')
176400          END;
176500        624 : (*EMP*)
176600          BEGIN IF READOBJ(OP) THEN
176700                BEGIN EMPTY(OP);
176800                      WRITELN(' OBJECT',OBJ,' MADE EMPTY')
176900                END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
177000          END;
177100        641 : (*INT*)
177200           BEGIN IF READOBJ(OP1) THEN
177300                 BEGIN
177400                  IF READOBJ(OP2) THEN
177500                  BEGIN
177600                   IF READOBJ(OP) THEN
177700                        INTERSECT(OP1, OP2)
177800                   ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
177900                  END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
178000                END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
178100          END;
       $
```

```
178200        595 : (*DIF*)
178300          BEGIN IF READOBJ(OP1) THEN
178400                BEGIN
178500                 IF READOBJ(OP2) THEN
178600                 BEGIN
178700                  IF READOBJ(OP) THEN
178800                       DIFF(OP1, OP2)
178900                  ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
179000                 END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
179100                END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
179200          END;
179300        642 : (*UNI*)
179400          BEGIN IF READOBJ(OP1) THEN
179500                BEGIN
179600                 IF READOBJ(OP2) THEN
179700                 BEGIN
179800                  IF READOBJ(OP) THEN
179900                       UNION(OP1, OP2)
180000                  ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
180100                 END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
180200                END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
180300          END;
180400        585 : (*ADD*)
180500          BEGIN IF READOBJ(OP) THEN
180600                BEGIN
180700                WRITELN(' ENTER TYPE OF ELEMENT');
180800                CH:=' '; WHILE CH=' ' DO READ(CH);
180900                WHILE NOT EOLN(INPUT) DO READ(ICH);
181000                CASE CH OF
181100                'P' : READPOINT(E, TRUE);
181200                'L' : READLINE(E, TRUE);
181300                'C' : READCONTOUR(E, TRUE);
181400                'T' : READTEXT(E, TRUE);
181500                'O' : READOBJEL(E, TRUE)
181600                END;
181700                ADD(E, OP)
181800                END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
181900          END;
182000        604 : (*DEL*)
182100          BEGIN IF READOBJ(OP) THEN
182200                BEGIN
182300                WRITELN(' ENTER TYPE OF ELEMENT');
182400                CH:=' '; WHILE CH=' ' DO READ(CH);
182500                WHILE NOT EOLN(INPUT) DO READ(ICH);
182600                CASE CH OF
182700                'P' : READPOINT(E, FALSE);
182800                'L' : READLINE(E, FALSE);
182900                'C' : READCONTOUR(E, FALSE);
183000                'T' : READTEXT(E, FALSE);
183100                'O' : READOBJEL(E, FALSE)
183200                END;
183300                DEL(E, OP)
183400                END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
183500          END;
              $
```

```
183600        621 : (*MEM*)
183700          BEGIN IF READOBJ(OP) THEN
183800                BEGIN
183900                WRITELN(' ENTER TYPE OF ELEMENT');
184000                CH:=' '; WHILE CH=' ' DO READ(CH);
184100                WHILE NOT EOLN(INPUT) DO READ(ICH);
184200                CASE CH OF
184300                'P' : READPOINT(E, FALSE);
184400                'L' : READLINE(E, FALSE);
184500                'C' : READCONTOUR(E, FALSE);
184600                'T' : READTEXT(E, FALSE);
184700                'O' : READOBJEL(E, FALSE)
184800                END;
184900                IF MEMBER(E, OP ) THEN
185000                WRITELN(' ELEMENT IN OBJECT ',OBJ)
185100                ELSE WRITELN(' ELEMENT NOT IN OBJECT ',OBJ)
185200                END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
185300          END;
185400        637 : (*TRA*)
185500          BEGIN IF READOBJ(OP1) THEN
185600                BEGIN
185700                 IF READOBJ(OP) THEN
185800                 BEGIN
185900                  CREATEMATRIX(TM);
186000                  TRAFO(OP1, TM)
186100                 END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
186200                END ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
186300          END;
186400        648 : (*WRI*)
186500          BEGIN IF READOBJ(OP) THEN
186600                  WRITEOBJ(OP)
186700                ELSE WRITELN(' OBJECT ',OBJ,' NOT DECLARED')
186800          END;
186900        623 : (*DIS*)
187000          BEGIN ENDOFD:=FALSE;
187100                PLOT11;
187200                WHILE NOT ENDOFD DO
187300                BEGIN
187400                 IF READOBJ(OP) THEN
187500                 BEGIN
187600                  WRITELN(' ENTER VX-VALUE');
187700                  READ(VX);
187800                  WRITELN(' ENTER VY-VALUE');
187900                  READ(VY);
188000                  WRITELN(' ENTER VZ-VALUE');
188100                  READ(VZ);
188200                  WRITELN(' ENTER DX-VALUE');
188300                  READ(DX);
188400                  WRITELN(' ENTER DY-VALUE');
188500                  READ(DY);
188600                  WRITELN(' ENTER DZ-VALUE');
188700                  READ(DZ);
188800                  VX:=VX-DX; VY:=VY-DY; VZ:=VZ-DZ;
      $
```

```
188900                  TM[1,1]:=1; TM[1,2]:=0; TM[1,3]:=0; TM[1,4]:=0;
189000                  TM[2,1]:=0; TM[2,2]:=1; TM[2,3]:=0; TM[2,4]:=0;
189100                  TM[3,1]:=0; TM[3,2]:=0; TM[3,3]:=1; TM[3,4]:=0;
189200            TM[4,1]:=-DX; TM[4,2]:=-DY; TM[4,3]:=-DZ; TM[4,4]:=1;
189300                  WRITELN(' PERSP/AXON?');
189400                  CH:=' '; WHILE CH=' ' DO READ(CH);
189500                  WHILE NOT EOLN(INPUT) DO READ(ICH);
189600                  IF CH='P' THEN PROJ3(VX, VY, VZ)
189700                            ELSE AXON3(VX, VY, VZ);
189800                  DISPLAY(OP, TM);
189900                  WRITELN(' MORE OBJECTS TO DISPLAY-YES/NO?');
190000                  CH:=' '; WHILE CH=' ' DO READ(CH);
190100                  WHILE NOT EOLN(INPUT) DO READ(ICH);
190200                  IF CH<>'Y' THEN ENDOFD:=TRUE
190300                 END ELSE WRITELN(' OBJECT NOT DECLARED')
190400                END; (*WHILE*)
190500                DEVEND
190600          END;
190700        606 : (*END*)
190800          BEGIN WRITELN(' END OF SESSION');
190900                ENDOFPR:=TRUE
191000          END
191100        END (*CASE*)
191200      END (*WHILE*)
191300 END.
      #
```