J.H. Louwsma

Constraints in Geo Virtual Reality applicaties

Inleiding

Geo-gerelateerde VR applicaties zijn er volop. Eén van de toepassingen van Geo-VR applicaties is het verbeteren van de communicatie over ruimtelijke plannen. Deze communicatie betreft het ontwerp proces, de presentatie en de interactie.

Het CGI is al een aantal jaren bezig met 3D applicaties voor ondersteuning bij ruimtelijke planvorming. Eén van de ontwikkelde applicaties is SALIX. SALIX is een VR simulatie programma voor landschapsarchitectonische ontwerpen. In het programma zijn groeisimulaties van beplantingsobjecten in een parkomgeving mogelijk. Het programma kan echter nog op een aantal punten verbeterd worden. Eén van deze verbeteringen betreft constraints.
Beplantingsobjecten (bomen en struiken) kunnen nu namelijk nog op bijvoorbeeld wegen en in het water geplaatst worden, wat niet overeenkomt met de werkelijke wereld. De objecten in het programma zouden beperkt (constraint) kunnen worden, zodat het programma beter aansluit bij de werkelijke wereld. Dit was de aanleiding om te zoeken naar mogelijkheden voor constrainted objecten in geo-VR, wat resulteerde in een afstudeeronderzoek.

De doelstelling van het onderzoek naar constraints in geo-VR applicaties luidde: definieer de beste manier om constraints in geo-VR omgevingen te implementeren. Allereerst is bekeken of bestaande geo-VR applicaties ook constraints bevatten. Aangezien er erg veel applicaties zijn, is dit onderzoek beperkt tot applicaties die lijken op SALIX. SALIX bestaat uit een DBMS voor de opslag van alle data, uit een Virtual Reality Modelling Language (VRML) omgeving voor de visualisatie en Java is gebruikt om uitgebreidere interactiemogelijkheden in de applicatie te implementeren. Er is dus gekeken naar applicaties die bestaan uit een DBMS, VRML en Java voor 3D visualisatie via internet of een andere VR omgeving voor visualisatie. Daarna is gezocht naar de beste manier om constraints in een geo-VR applicatie te implementeren.

Begrip constraint en soorten constraints

Er bestaan al heel veel geo-VR applicaties die opgebouwd zijn uit bovengenoemde componenten. Een heel aantal daarvan zijn bekeken, waarbij er gezocht is naar constraints. Echter geen van de bekeken applicaties bevatte constraints. Het was dan ook nodig eerst een definitie van constraints in geo-VR te geven, gevolgd door het zoeken naar de beste manier van implementatie.

De definitie van een constraint in geo-VR is afgeleid uit de algemene definitie van een constraint en luidt:

Een beperkte of gedwongen **relatie**
tussen **objecten**.

In deze definitie staan de begrippen relatie en objecten centraal. Als de verschillende soorten relaties tussen objecten bekend zijn, kunnen hieruit dus ook de verschillende soorten constraints afgeleid worden.
Een lijst van verschillende soorten relaties tussen objecten, en dus ook de verschillende soorten constraints, is opgesteld. De mogelijke relaties zijn:

- ruimtelijke relaties, die onderverdeeld kunnen worden in:
  - topologische;
  - metrische;
  - ordelijke relaties.
- temporele relaties;
- kwantitatieve relaties;
- thematische relaties.

Een lijst van voorbeeldconstraints is te zien in tabel 1.

SALIX is in dit onderzoek gebruikt als voorbeeld applicatie voor een implementatie van constraints. Voor deze implementatie zijn de voorbeeld constraints uit tabel 1 gebruikt. Er is getracht een representatieve lijst samen te stellen, waarbij voor elk type constraint een voorbeeld gedefinieerd is, echter niet alle constraints zijn goed toepasbaar voor SALIX (temporele en ordelijke constraints). Het laatste voorbeeld in tabel 1 is een complexe constraint, dit is niets anders dan een samenstelling van twee of meer enkelvoudige constraints (die maar één relatie bevatten).

Type constraint	Constraints voor de implementatie in SALIX
Ruimtelijk topologisch	1. Struiken mogen nooit in water staan
Ruimtelijk metrisch	2. Een struik moet altijd ten zuiden van een boom geplaatst worden. 3. Bomen moeten altijd > 1 meter van wegen en paden geplaatst worden
Kwantitatief	4. Er moeten minstens 10 bomen op een specifiek (nader te def.) grondvlak staan
Thematisch	5. Een struik mag een pad of weg niet overlappen.
Complex	6. De afstand tussen bomen in water is altijd > 8m EN de afstand tussen een boom in het water en de rand van het water moet altijd < 0,5 meter zijn EN de enige soort boom die in het water mag staan is een ´QueRob´.

Tabel 1 Constraints die in SALIX geïmplementeerd zijn

Constraints kunnen gedefinieerd worden tussen twee objecten, maar ook als een deel van een objectbeschrijving. Zo kan bijvoorbeeld de constraint ‘object A moet altijd binnen object B liggen’ ook geschreven worden als een deel van de beschrijving van object A; ´moet altijd binnen object B liggen´ en als deel van de beschrijving van object B; ´moet altijd object A bevatten´. Deze opsplitsing in delen van objectbeschrijvingen kan helpen bij het nagaan of alle gedefinieerde constraints in een applicatie niet conflicteren door ze in te vullen in de zogenaamde kruis relatie tabel, zie tabel 2.

	Object A	Object C
Object B	altijd in bevat altijd	raakt altijd altijd > 4m van

Tabel 2 Kruis relatie tabel waarin constraints als deel van de objectbeschrijvingen gegeven worden

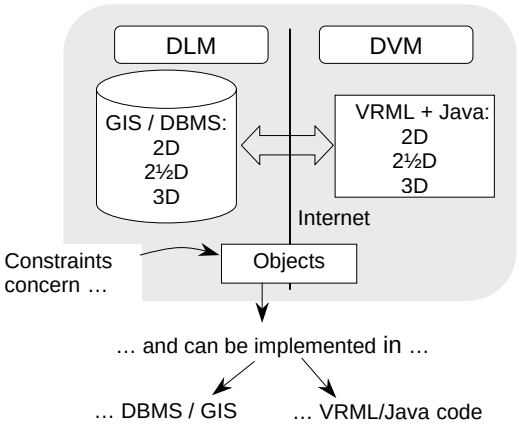
In tabel 2 is te zien dat de constraint van object A niet conflicteert met de constraint van object B. De constraint van object C conflicteert echter wel met de constraint van object B. Eén van beide moet dus aangepast worden om een sluitende lijst van constraints te krijgen voor de implementatie in een geo-VR applicatie. Hoe langer de lijst van constraints, hoe groter de kans op conflicten, maar anderzijds kunnen er met een aantal constraints ook bepaalde patronen gedefinieerd worden (bijvoorbeeld m.b.v. afstanden en richtingen tussen objecten).

Implementatie mogelijkheden van constraints

Na het definiëren van een lijst van constraints voor een applicatie moet nagedacht worden over de beste manier om constraints te implementeren. Applicaties die bestaan uit een DBMS, VRML en Java kunnen opgedeeld worden in:

1. visuele omgeving;
2. database omgeving.

De visuele omgeving kan vergeleken worden met de client side van een client-server applicatie en de database omgeving kan vergeleken worden met de server side. Ook kan de visuele omgeving gezien worden als het digitale visualisatie model (DVM) dat een *visualisatie* van de werkelijkheid geeft en de database omgeving als het digitale landschapsmodel (DLM) dat alle data bevat om het landschap te *beschrijven.* In deze DLM-DVM onderverdeling moeten de constraints een plaats krijgen. De constraints kunnen dus zowel in het DLM geïmplementeerd worden (opslag in DBMS) als in het DVM (opslag in VRML/Java) (zie figuur 1).



Figuur 1 De plaats waar constraints geïmplementeerd kunnen worden in een applicatie bestaande uit een DLM en DVM

Voor beide omgevingen volgen nu de overwegingen en mogelijkheden.

Implementatie mogelijkheden visuele omgeving

Ervan uitgaande dat de DVM opgebouwd is uit VRML en Java (wat het geval is voor SALIX), is een korte introductie van VRML op zijn plaats. VRML is een modelleer taal om 3D objecten te beschrijven.

De objecten in het VRML model zijn nodes. Nodes bevatten velden waarin de eigenschappen van de node gedefinieerd worden (zoals de kleur en de geometrie). Er moet na elke verandering in het VRML model (in SALIX is dit na het verwijderen of (ver-)plaatsen van objecten) getest worden of nog voldaan wordt aan de constraints. De nodes die hier het meest geschikt voor lijken, zijn de Collision node en de TouchSensor node.

De Collision node signaleert een botsing tussen de avatar (de representatie van de persoon die door het VRML model wandelt) en de geometrie van de collision node. Een botsing tussen twee geometrieën onderling wordt dus niet gesignaleerd, terwijl dit juist nodig is om constraints te implementeren (vb. een *struik* mag niet in het *water* geplaatst worden).

De TouchSensor node detecteert wanneer de gebruiker naar de geometrie van de TouchSensor node wijst (met bv een muis). Na het aanklikken van de geometrie, wordt een TouchTime gegenereerd (het tijdstip van het aanklikken). Deze TouchTime kan een andere node activeren. Deze node zou een Javascript kunnen bevatten, waar de constraints in geïmplementeerd kunnen worden. De code om te checken of aan de constraints voldaan wordt, moet echter helemaal geprogrammeerd worden zonder gebruik te kunnen maken van bestaande functies en operaties. Dit in tegenstelling tot een DBMS. Verder worden de constraints niet op een eenvoudig toegankelijke centrale plaats opgeslagen. De implementatie van constraints kan wellicht efficiënter in het DLM (DBMS).

Implementatie mogelijkheden DBMS

Alle DBMSen hebben de mogelijkheid om 'integrity constraints' te implementeren. De integrity constraints in DBMSen zijn meer omvattend dan de constraints in geo-VR. De volgende categorieën van integrity constraints zijn te onderscheiden:

- domain constraints (zijn voor dit onderzoek niet van belang);
- general constraints;
- base table constraints.

De general constraints, oftewel assertions, zijn expressies die nooit 'false' mogen opleveren en hebben de volgende syntax:

```
CREATE ASSERTION <assertion_naam>
CHECK <constraint_body>.
```

Hierbij is de constraint body een SQL expressie, waarin bestaande functies en operaties gebruikt kunnen worden (zoals het maken van een buffer of berekenen van afstanden tussen geometrieën). Een voorbeeld van een assertion is gegeven in figuur 2.

```
CREATE ASSERTION constraint_1 CHECK
(NOT EXISTS    (SELECT * FROM prcv_treesrd_point t, prcv_gvkrd_poly g
                WHERE t.kind = ´bush´
                AND g.descript = ´water´
                AND SDO_RELATE (g.geom, t.geom, ´mask=inside, querytype=window´)=´TRUE´))
```

Figuur 2 Voorbeeld van een assertion.
Hier is constraint 1 uit tabel 1 (struiken mogen nooit in water staan) gedefinieerd als assertion. Prcv_treesrd_point is de tabel met de locaties van alle bomen en struiken en prcv_gvkrd_poly is de tabel met de geometrie van het grondvlak

De base table constraints zijn gerelateerd aan een tabel en er zijn drie soorten base table constraints:

- Candidate key definition: {PRIMARY KEY|UNIQUE};
- Foreign key definition: FOREIGN KEY;
- Check constraint definition: CHECK.

De check constraint definitie kan voor het implementeren van constraints in dit onderzoek gebruikt worden. De syntax van een check constraint is:

```
CREATE TABLE <table_name> (´kolom_1´
´data_type_voor_kolom_1´,
 ´kolom 2´ ´data_type_voor_kolom_2´, ... )
CONSTRAINT <constraint_name> CHECK
(constraint_body).
```

General en base table constraints kunnen dezelfde inhoud bevatten. Het voordeel van een general constraints is dat deze niet aan een tabel gerelateerd hoeft te worden. De constraints voor geo-VR zouden dus het beste d.m.v. general constraints geïmplementeerd kunnen worden. Echter, de bestaande DBMSen ondersteunen geen general constraints en ook het alternatief, de base table check constraint, is beperkt. De check kan namelijk geen subqueries bevatten. DBMSen bieden andere manieren om complexe constraints te implementeren. Voor Oracle wordt bijvoorbeeld de ontwikkeltool Custom Development Method (CDM) RuleFrame gebruikt om complexere constraints te implementeren. Zulke tools zijn in het kader van dit onderzoek niet verder onderzocht.

Eenvoudige constraints kunnen dus wel geïmplementeerd worden, maar de complexere constraints kunnen niet met behulp van de in de

literatuur bestaande integrity constraints geïmplementeerd worden. Voor de complexere constraints kunnen echter wel database triggers en procedures gebruikt worden.

In dit onderzoek is gebruik gemaakt van Oracle als voorbeeld DBMS voor het zoeken naar mogelijkheden en voor het implementeren van een aantal voorbeeld constraints in SALIX. De componenten van de database triggers in Oracle zijn genoemd in tabel 3.

Component	Syntax
trigger name	`create [or replace] trigger < trigger name>`
trigger time point	`before | after`
trigger event(s)	`insert or update [of <column(s)] or delete on <table>`
trigger type (optional)	`for each row`
trigger restriction (only for for each row triggesr)	`when (condition)`
trigger body	`<PL/SQL block>`

Tabel 3 Componenten van database triggers in Oracle

Een trigger wordt voor of na een insert, update en/of een delete commando op de trigger tabel doorlopen. De trigger body is een PL/SQL blok en kan alle functies en operaties van Oracle bevatten. Wanneer niet aan een trigger voldaan wordt, volgt een impliciete rollback in de database. De mogelijkheden voor het implementeren van de verschillende soorten constraints zijn als volgt:

- Voor *ruimtelijk topologische* constraints kan de `SDO_RELATE` operatie gebruikt worden. Hiermee zijn alle topologische relaties te definiëren.
- Voor *ruimtelijk metrische* constraints kan de `SDO_WITHIN_DISTANCE` operatie gebruikt worden voor de afstanden en voor richtingen kunnen trigonometrische operaties (zoals sin, cos en tan) gebruikt worden.
- Voor de *ruimtelijk ordelijke* constraints kan de `SDO_RELATE` operatie gebruikt worden om te definiëren dat sommige geometrieën altijd onderdeel zijn van andere geometrieën (bv. inside).
- De *temporele* constraints kunnen geïmplementeerd worden, gebruik makend van tijdstip en data.
- De *kwantitatieve* constraints kunnen geïmplementeerd worden met de volgende SQL functie: `SELECT count(*) FROM <table_name>`.
- De *thematische* constraints kunnen geïmplementeerd worden met het standaard `SELECT-FROM-WHERE` SQL commando met in de `WHERE` component de specificatie van de thematische attributen.

Vergelijking visuele omgeving vs. DBMS

Database triggers en procedures bieden de meest efficiënte implementatiemogelijkheden voor constraints in geo-VR. Constraints worden centraal opgeslagen in de DBMS en bestaande functies en operaties in de DBMS kunnen gebruikt worden binnen de triggers en de procedures. Dit scheelt aanzienlijk in de hoeveelheid programmeerwerk in vergelijking met een implementatie gebruikmakend van Java.

Verder is het wenselijk dat constraints aangepast kunnen worden. Een gebruiker moet bijvoorbeeld de mogelijkheid hebben om constraints te activeren of deactiveren of zelfs nieuwe constraints te maken of bestaande constraints aan te passen. Ervan uitgaande dat de gebruiker niet een frequent GIS gebruiker is, is het aanpassen en het creëren van nieuwe constraints lastig te implementeren met zowel Java als met triggers en procedures. Het activeren en deactiveren van constraints is daarentegen wel makkelijk te implementeren als gebruik wordt gemaakt van triggers in een DBMS. In Oracle kunnen triggers namelijk afzonderlijk geactiveerd (enable) en gedeactiveerd (disable) worden. Wordt gekozen voor een implementatie met behulp van Java dan is dit veel complexer.

Implementatie in SALIX

Na het zoeken naar mogelijkheden voor het implementeren van constraints in geo-VR is besloten om gebruik te maken van de database triggers en procedures in Oracle voor de implementatie van constraints in SALIX. Omdat assertions redelijk eenvoudig te definiëren zijn en niet gerelateerd hoeven te worden aan tabellen, zijn de constraints in tabel 1 allereerst gedefinieerd als assertions. Deze assertions zijn omgeschreven naar PL/SQL zodat ze in de triggers en procedures gebruikt konden worden.
Er is gekozen om een ´before each row´ trigger te gebruiken. Alleen binnen een before each row trigger kunnen namelijk de nieuwe (en oude) attribuutwaarden verkregen worden van het betrokken object. Het betrokken object is het object in het insert, update of delete statement, dat de trigger aangeroepen heeft. Als een object verplaatst of geplaatst wordt in een model en deze verandering opgeslagen wordt in de DBMS, moet getest worden of nog aan alle constraints voldaan wordt. Voor deze test zijn de nieuwe attribuutwaarden nodig van het betreffende object.

In de before each row trigger worden de nieuwe attribuutwaarden geselecteerd en opgeslagen in een tijdelijke tabel (involved_object), die aangeroepen kan worden in alle triggers en procedures, waarna gekeken wordt welke constraints voor dat object van belang zijn. Voor de constraints die van belang zijn wordt één voor één getest of na de verandering in het VR-model nog aan de constraints voldaan wordt.

Voor elke constraint is een procedure gemaakt, die allemaal verzameld zijn in een package voor een overzichtelijke opslagstructuur. Zo wordt bijvoorbeeld in de before each row triger met ´pck_salix.pr_topology_c1´ de procedure pr_topology_c1 aangeroepen Alle procedures zijn opgeslagen in de package met de naam pck_salix.. De procedure pr_topology_c1 test de constraint: ´Struiken mogen nooit in water staan´ en de PL/SQL code van deze procedure is afgeleid uit de assertion van tabel 1. De code van de procedure ziet er als volgt uit:

```
PROCEDURE pr_topology_c1
IS
   description varchar2(10);
   bush_in_water EXCEPTION;
BEGIN
   select g.descript INTO description
   from involved_object i, prcv_gvkrd_poly g
   where sdo_relate(g.geom, i.geom,
'mask=CONTAINS, querytype=window')='TRUE';
   IF description = 'water' THEN
      raise bush_in_water;
   ELSE DBMS_OUTPUT.PUT_LINE('1: the bush is
not placed in water');
   END IF;
EXCEPTION
   WHEN bush_in_water THEN
      raise_application_error (-20001,
      '1: The bush is placed inside water,
but a bush may never be placed in water.
      place the bush on another location.');
END pr_topology_c1;
```

Feedback

Na het creëren van de triggers en procedures moet de gebruiker van de applicatie geïnformeerd worden over de geldende constraints en wanneer deze worden overtreden. Met de procedures wordt al tekst als output gegeven. Deze tekst moet alleen nog naar de visuele omgeving van de applicatie gestuurd worden. Binnen een Java omgeving is daarvoor (nadat een SQL query naar de database gestuurd is) het volgende commando nodig:

```
catch(SQLException e)
{
System.out.println("exception: " +
e.getMessage());
}
```

De getMessage() vangt hierbij het nummer van de foutmelding en de beschrijving op. Tekstuele feedback bij het starten van de applicatie of visuele feedback is niet geïmplementeerd in SALIX, maar zou de gebruiker nog beter informeren. Vooral visuele feedback is wenselijk, een plaatje zegt immers meer dan duizend woorden. Voor dit onderzoek is de visuele feedback echter buiten beschouwing gelaten.

Conclusie

Constraints zijn beperkte of gedwongen relaties tussen objecten en de volgende typen constraints zijn te onderscheiden:
Ruimtelijke constraints, die weer onderverdeeld kunnen worden in:
topologische;
metrische;
ordelijke constraints.
temporele constraints;
kwantitatieve constraints;
thematische constraints.

De constraints zijn te implementeren in de visuele omgeving of in de DBMS van een geo-VR applicatie. De visuele omgeving biedt mogelijkheden doordat Java gebruikt kan worden. DBMSen bieden echter efficiëntere manieren om constraints te implementeren. Database triggers en procedures kunnen gebruikt worden voor de implementatie van complexe constraints. De voordelen van deze manier van implementeren ten opzichte van een implementatie met Java zijn:
de constraints worden opgeslagen op een centrale plaats;
binnen de triggers en procedures kunnen bestaande functies en operaties gebruikt worden;
het is mogelijk om triggers afzonderlijk te activeren en te deactiveren.
Het gebruik van general constraints (assertions) zou eenvoudiger zijn, deze worden echter niet ondersteund door DBMSen.

Feedback om de gebruiker te informeren over de constraints is belangrijk. Visuele feedback geeft in één oogopslag meer informatie en is dus wenselijk, maar is veel lastiger te realiseren dan tekstuele feedback. In SALIX wordt alleen achteraf tekstuele feedback gegeven (nadat het VR model aangepast is en dit opgeslagen wordt in de DBMS).

Dit onderzoek is beperkt tot geo-VR applicaties bestaande uit een DBMS, VRML en Java. Er bestaan veel meer applicaties met een andere structuur die wellicht andere implementatie mogelijkheden bieden. Naar deze andere implementatie mogelijkheden zou zeker eens naar

gekeken kunnen worden.

Verder zouden DBMSen en hun mogelijkheden om complexe constraints te implementeren nader bekeken moeten worden. Vooral een nader onderzoek naar de mogelijkheden van ontwikkeltools (zoals Oracle´s CDM RuleFrame) om complexe constraints te implementeren is aan te bevelen.