Geleidelijke en geanimeerde voor vario-schaal kaarten via

Zo'n vijf jaar geleden is in Geo-Info het concept van vario-schaal geo-informatie beschreven (Van Oosterom en Meijers, 2012). In dit eerdere artikel werd de eerste, echt geleidelijke vario-schaal structuur gepresenteerd: een delta schaal geeft een delta in de kaart (en hoe kleiner de delta schaal hoe kleiner de delta kaart). De afgelopen vijf jaar is veel R&D verricht om het concept van vario-schaal geo-informatie te realiseren: het ontwikkelen van prototypen en testen met echte data. In het kader van het Open Technologieprogramma (OTP van STW, Stichting Technische Wetenschappen) project 11185 'Vario-scale geo-information' is er de afgelopen jaren veel vooruitgang geboekt. De belangrijkste resultaten worden in een serie beknopte artikelen behandeld. Dit is het zesde en laatste artikel in de serie.

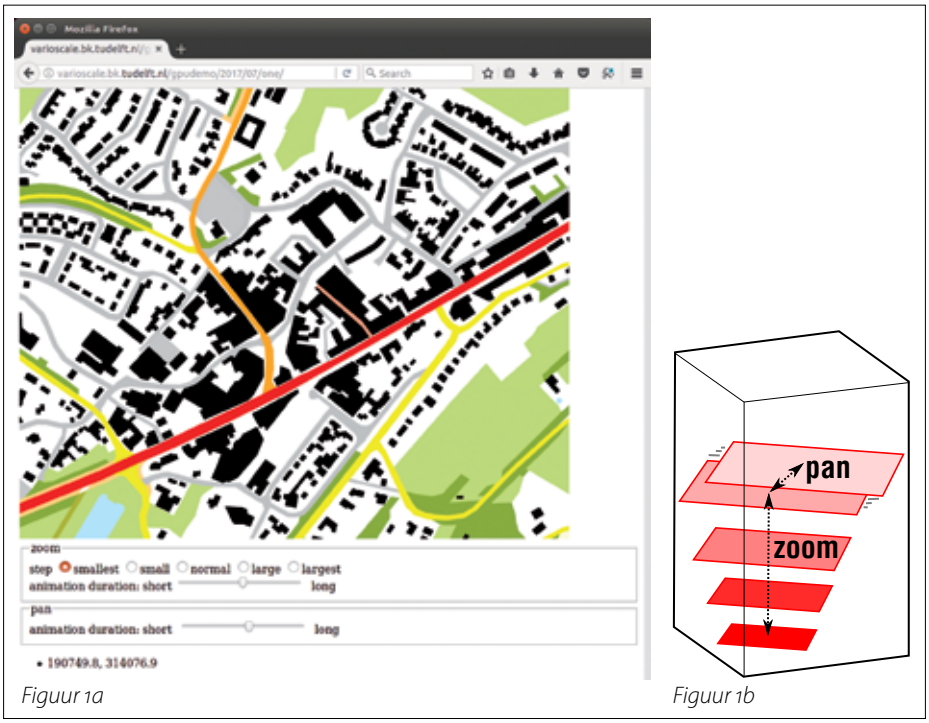
Door Martijn Meijers en Peter van Oosterom

De ruimte schaal kubus ('Space Scale Cube', SSC) slaat het resultaat van generalisatie operaties op, met bijbehorende geleidelijke overgangen. De derde dimensie wordt hierbij gebruikt om geleidelijke overgangen tussen objecten op verschillende kaartschalen geometrisch te beschrijven, waarbij ook de grenzen van de objecten geleidelijk kunnen veranderen (zie het vijfde artikel in deze serie). Veel gebruikte kaartgeneralisatie operaties passen in deze structuur. Dit 3D-model om 2D-kaarten uit af te leiden kan zodoende direct worden gebruikt in een (mobile) client, waar tegenwoordig krachtige grafische hardware beschikbaar is. Dit artikel beschrijft een architectuur voor het gebruik van geleidelijke SSC-data over het web. Als eerste bespreken we hoe data in blokken worden georganiseerd, bevraagd en gebruikt voor een efficiënte client-server aanpak. Daarna laten we zien hoe onze implementatie van geleidelijk en geanimeerd zoomen is gerealiseerd. De architectuur is geïmplementeerd als prototype (figuur 1(a)). Het prototype is te vinden op de website varioscale.bk.tudelft.nl/ (onder WebGL demo).

Een client voor op het web

Ons eerdere (beperkte) gebruikersonderzoek, met een client op een desktop omgeving, inclusief gebruikmaking van de grafische hardware (GPU), liet zien, dat gebruik van vario-schaal data staat of valt met de mogelijkheden voor goede gebruikersinteractie. Er is gekozen voor disseminatie via het web met een 'fat' client, vanwege de mogelijkheden die dit platform biedt (door de beschikbare en krachtige hardware). Een 'fat' web client biedt de volgende voordelen (met name door het renderen van vector data):

- WebGL is beschikbaar binnen alle grote browser implementaties en laat de noodzaak vervallen om een plug-in of een specifieke (desktop) applicatie te installeren. WebGL is ook in mobiele browsers beschikbaar (gebruik op mobiele telefoon of tablet);
- Interactief is de stijl (bijvoorbeeld kleuren) van het kaartbeeld te wijzigen;



Figuur 1a Figuur 1b

Figuur 1 - (a) Het gerealiseerde prototype, met daaronder de kaartopties om de animatie tijdens de gebruikersinteractie te beïnvloeden. (b) De gebruiker kan het snijvlak in de SSC beïnvloeden met schuif- en zoomacties. Het afgeleide kaartbeeld wordt geanimeerd aan de gebruiker getoond (door snijvlakposities te interpoleren tussen begin en eind).

interactie het web

- Arbitraire rotaties van het kaartbeeld worden mogelijk, als ook eventuele perspectivische weergave van het kaartbeeld;
- Interactie met objecten is mogelijk, bijvoorbeeld het op laten lichten van een object nadat een gebruiker erop geklikt heeft.

Geleidelijke SSC- data in client-server architectuur

Het interactief omgaan (schuiven en zoomen) met 2D-kaarten gebeurt in onze aanpak door het doorsnijden van de ruimte-schaal kubus, waarbij de gebruiker de plaats en grootte van het snijvlak kan beïnvloeden (figuur 1(b)). Het resultaat van deze doorsnijdingsoperatie kan in real-time gevisualiseerd worden door efficiënt gebruik te maken van de GPU. Om deze aanpak te laten werken, dienen de SSC-data op de client beschikbaar gemaakt te worden. Geleidelijke overgangen kunnen met behulp van een animatie aan de gebruiker worden getoond, zodat het gevoel van geleidelijkheid bij kaartgebruik nog beter bij de gebruiker overkomt.

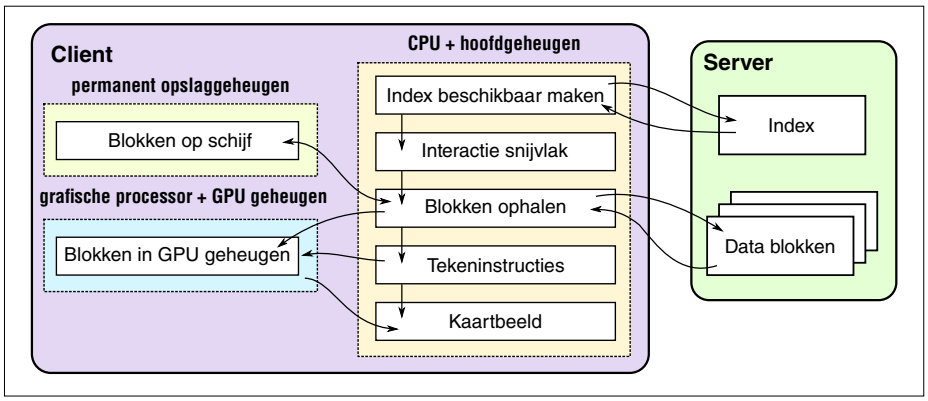
Architectuur op basis van data blokken

Figuur 2 toont op hoofdlijnen de architectuur met zijn componenten:

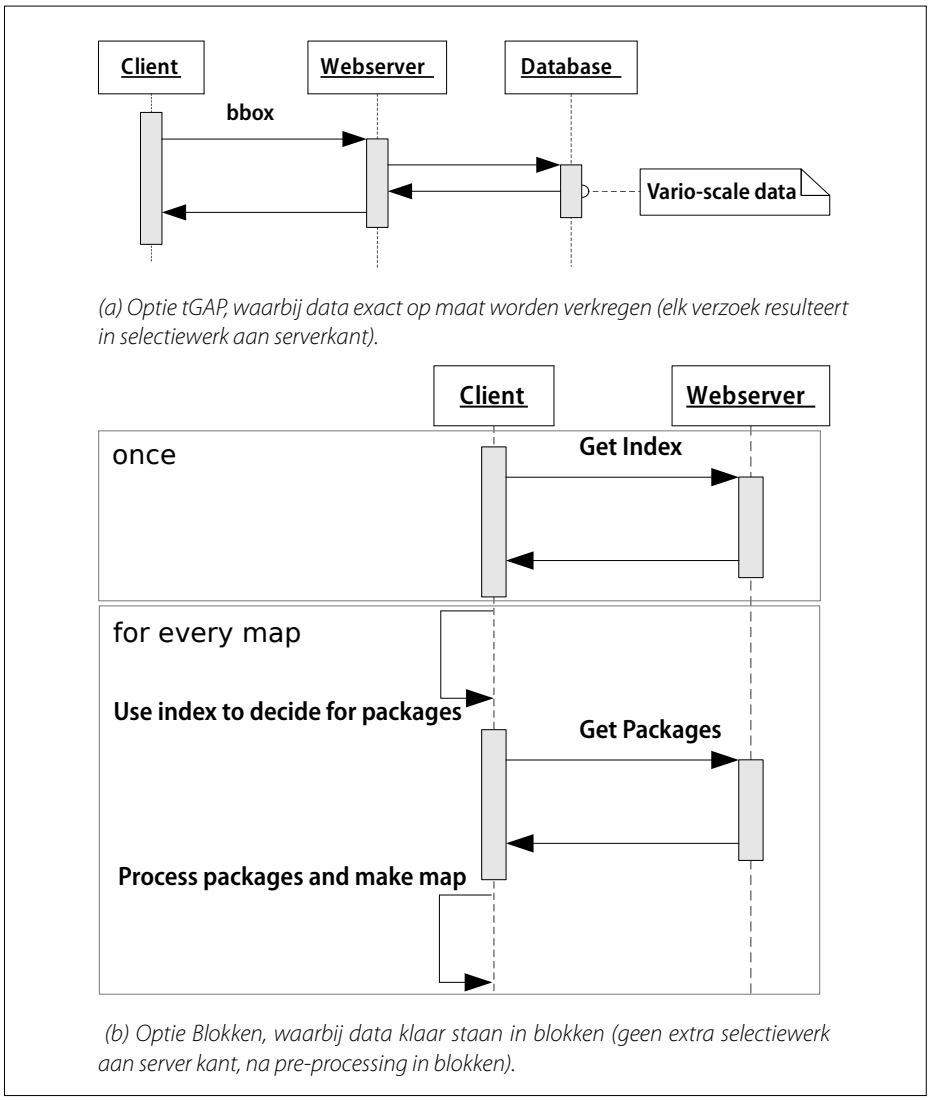
1. de vario-schaal server, met blokken met data en een bijbehorende index structuur;
2. de web client: met beschikking over hoofdgeheugen, een CPU, geheugen op de grafische kaart en de grafische processor, GPU en ook de mogelijkheid om van permanent opslaggeheugen gebruik te maken (zoals bijvoorbeeld een SSD kaart in mobiele telefoon).

De web client is gemaakt met behulp van Javascript en WebGL. Javascript maakt het mogelijk om logica aan webpagina's toe te voegen en WebGL biedt de mogelijkheid om gebruik te maken van de GPU.

In eerder werk (tweede artikel van deze serie) is een protocol beschreven voor het opvragen van vario-schaal data (figuur 3(a)). De client stuurt hierbij een bounding box naar de server. Vervolgens moet de server bepalen welke vario-schaal data de client exact nodig heeft om de kaart te kunnen tekenen (waarbij de volledige tGAP structuur aan de serverkant beschikbaar is). Bij de SSC-aanpak is dit protocol aangepast, waarbij van tevoren datablokken op



Figuur 2 - Overzicht van de architectuur. Door gebruik te maken van blokken kan data efficiënt worden overgestuurd, hetgeen onder andere hergebruik aan de clientkant mogelijk maakt.



Figuur 3 - Optie tGAP (ophalen exacte vario-schaal data) versus Optie Blokken (ophalen data in pakketten).

de server worden klaargezet die de client door middel van de beschikbaarheid van een index structuur kan aanvragen (figuur 3(b)).

Zodra de client de indexstructuur heeft ontvangen, kan deze worden gebruikt om nieuwe blokken te vragen (tijdens schuiven en zoomen). Verder kan de indexstructuur ook worden gebruikt om te voorkomen dat een datablok meerdere keren wordt overgehaald, door te administreren welke status elk blok heeft (bijvoorbeeld: 'nog niet beschikbaar', 'in aanvraag', 'ontvangen', 'ontvangen en beschikbaar gemaakt op de GPU'). Eveneens wordt de taak van de server eenvoudiger (schaalbaarder bij meerdere gebruikers tegelijkertijd) en kan een deel van de datablokken voor gebruik eenmalig overgehaald worden, zodat later, zonder netwerkverkeer, ook met de kaart geïnteracteerd kan worden. Opgevraagde blokken moeten als laatste stap beschikbaar gemaakt worden in het geheugen van de GPU, zodat de data nu gebruikt kunnen worden als er teken-instructies op de GPU worden afgevuurd (door middel van zogenaamde 'shader' programma's).

Om datablokken te verkrijgen, is het nodig om de volgende stappen uit te voeren: van tevoren moet een goed gevulde tGAP structuur beschikbaar zijn (met alle uitdagingen van automatische generalisatie). Vervolgens vindt een vertaling plaats van de tGAP structuur naar een SSC (met een expliciete 3D-beschrijving op basis van polyhedrons, 'boundary representation'). Deze 3D-representatie beschrijft geleidelijke overgangen die de gebruiker op zijn scherm te zien krijgt. Deze representatie van 3D-polyhedrons wordt omgezet naar een structuur die geschikt is voor direct gebruik op de grafische hardware, gebaseerd op driehoeken. Als laatste stap wordt deze dataset met 3-driehoeken in blokken verdeeld en wordt een indexstructuur geproduceerd (welke blokken zijn er, wat is hun ruimtelijke extent, eventueel met de hoeveelheid data van elk blok).

Datablokken en bijbehorende indexstructuur maken

Hoe moeten de blokken gemaakt worden, welke eisen stellen we aan ze en hoe moet de indexstructuur eruitzien? In theorie zijn er meerdere manieren mogelijk om datablokken te maken:

1. We kunnen de 3D-objecten opknippen als ze overlappen met een stukje ruimte (ruimteopdelende aanpak, bijvoorbeeld gebruik van een Octree);
2. We kunnen de 3D-objecten in zijn geheel intact laten en de objecten die dichtbij elkaar liggen vervolgens groeperen (objectopdelende aanpak, bijvoorbeeld door een R-tree te gebruiken).

Verder hebben we voor de blokken en de indexstructuur een aantal wensen:

a. Geen (of weinig) introductie van extra geometrie door de introductie van de blokken, liefst willen we ook object geometrie niet opknippen over meerdere blokken.
b. Blokken moeten ongeveer dezelfde hoeveelheid geometrie hebben (dit leidt namelijk tot het hebben van ongeveer dezelfde opslaggrootte). Zodra een blok aangevraagd wordt, is dan duidelijk hoeveel netwerkverkeer nodig is. Daarnaast is eenzelfde blokgrootte handig, omdat de blokken in het geheugen van de GPU moeten worden opgeslagen. Als deze blokken allemaal even groot zijn, kan het geheugen daar zonder fragmentatie goed gebruikt worden.
c. Blokken moeten compact zijn en niet één lange zijde hebben bijvoorbeeld, want het gevolg is dat een blok dan voor veel visualisaties nodig is.

Voor de indexstructuur, is het fijn als:

a. Bevragingen voor benodigde blokken op de client in real time mogelijk zijn (antwoord binnen aantal milliseconden).
b. De structuur ook in delen over te halen is (als er datablokken voor de hele wereld beschikbaar zijn, wordt ook de indexstructuur heel groot).

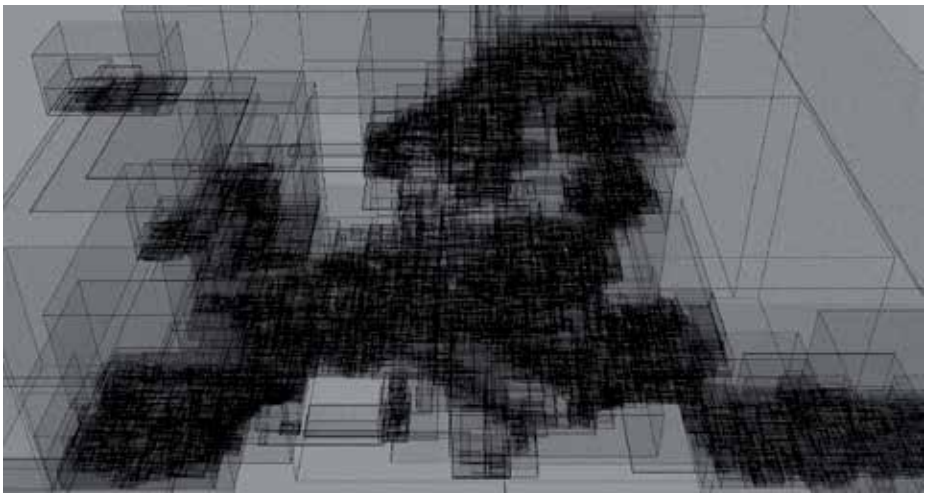
Een eerste optie om blokken te maken is een Octree structuur gebruiken (onderzocht door Mattijs Driel en Yueqian Xu). Dit leidt echter tot een explosie van data: testen voor een gebied van 9x9 km laten zien dat 40 MB aan opslagruimte nodig is voor één blok (dus zonder de SSC op te splitsen). Als de SSC wel opgesplitst wordt over 1135 blokken, hebben deze blokken samen een totale grootte van 239 MB. Zes keer meer data, alleen doordat de objecten opgeknipt moeten worden!

Door de geïntroduceerde redundantie lijkt deze structurering heel veel op een aanpak met meerdere kaartlagen. Elk blok met data heeft namelijk ook een bodem nodig, anders werkt de rendering aanpak niet (en kijk je als het ware door het blok met data heen). Hiervoor worden driehoeken die op de rand liggen opgeknipt en ook op de bodem van elk Octree blok worden driehoeken toegevoegd. Er moet dus worden geconstateerd dat deze aanpak het kind met het badwater weggooit.

Een andere poging om blokken te maken, is uitgewerkt door Adrie Rovers (figuur 4 geeft een voorbeeld van de gemaakte blokken). Zijn uitgangspunt is dat de 3D-objecten heel moeten blijven. Elk object houdt hiermee zijn eigen representatie compleet (inclusief bodem). Hierdoor wordt de redundantie van de Octree voorkomen. Een ruimte vullende curve kan gebruikt worden om groepen van de objecten te maken: elk object wordt benaderd door een punt in de 3D-ruimte-schaal kubus (x, y, schaal). Vervolgens wordt gekeken wat de locatie is van dit punt op de gekozen ruimte vullende curve. Door deze aanpak komen objecten op dezelfde locatie en voor gebruik op dezelfde schaal hopelijk bij elkaar in hetzelfde blok te liggen. Adrie Rovers' afstudeerwerk heeft laten zien, dat het tunen van deze aanpak vrij veel experimenteren vergt (hoe behandel je bijvoorbeeld de lengte van de schaaldimensie in relatie tot de lengte van een ruimtedimensie) en dat ook objecten die over veel schalen beschikbaar zijn, de organisatie van de objecten in goede groepen bemoeilijken.

Geanimeerd zoomen en pannen met de blokken structuur

Nu er blokken met data aan de client kant beschikbaar zijn, kunnen we de visualisatie en interactiemogelijkheden verder uitwerken. De manier waarop de eindgebruiker het



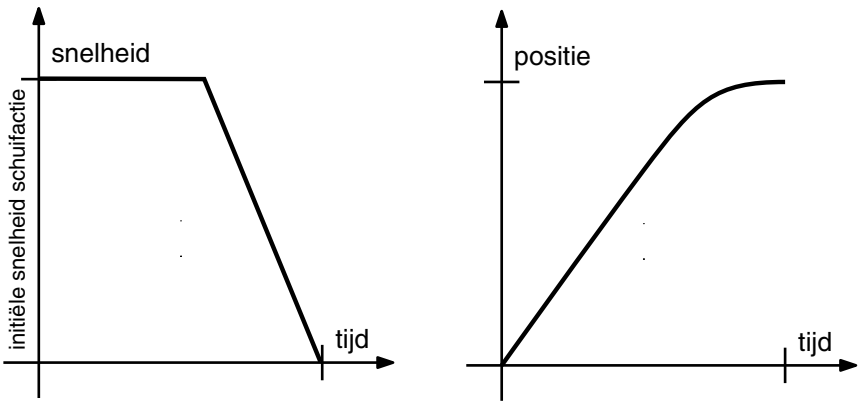
Figuur 4 - Visualisatie van datablokken voor een dataset met CORINE Europese landcover data, waarbij objecten niet opgeknipt worden.

doorsnijdingsvlak door de SSC kan bewegen, geeft de gebruiker een soepele en geleidelijke indruk bij het kaartgebruik. De Internetbrowser, die Javascript uitvoert, beschikt over een zogenaamde 'event loop'. De gebruiker kan acties doen en hiermee 'events' genereren. Concreet betekent dit, dat een gebruiker met de muis het systeem een instructie kan geven, bijvoorbeeld door middel van een muisklik of een klik van het muiswiel. Hiermee kan het doorsnijdingsvlak worden verplaatst en kunnen er andere acties worden gestart, bijvoorbeeld ophalen van data vanaf de server.

Zo kan een gebruikersactie ervoor zorgen dat er vervolgens een tekenactie op de GPU (grafische processor) plaatsvindt. Gebruikersacties (verplaatsing van het snijvlak) en tekenacties hoeven hierbij niet één-op-één aan elkaar gekoppeld te worden. Er kan bijvoorbeeld worden ingesteld, dat elke 1/60 seconde een tekenactie plaatsvindt voor een bepaalde tijdsduur, nadat de gebruiker de actie uitgevoerd heeft. Dit leidt ertoe dat het systeem een animatie van het kaartbeeld toont (er wordt als het ware een filmpje afgespeeld met tijdelijke kaartbeelden). Een beschrijving van de animatie wordt gegeven in de vorm van het huidige startpunt op de kaart (x_s, y_s, $schaal_s$), het gewenste eindpunt (x_e, y_e, $schaal_e$), de starttijd en de gewenste duur van de animatie. Een klik van het muiswiel leidt tot het aanvragen van blokken en het in werking stellen van een animatie die de kaart laat in- of uitzoomen. De zoomactie die geïmplementeerd is, vergroot (of verkleint) het kaartbeeld rond de plek waar de muis gepositioneerd is. Hierdoor kan de gebruiker eenvoudig rond een punt inzoomen. Om het aanvragen van data en tekenen met de GPU niet te veel te laten interfereren, kan in Javascript gebruik worden gemaakt van zogenaamde web workers. Een web worker start een tweede proces op binnen de internetbrowser, dat verantwoordelijk gemaakt kan worden voor bijvoorbeeld data overdracht. Voordeel is dat de client hierdoor interactiever blijft (omdat ontvangst van de data in parallel kan worden afgehandeld met de tekeninstructies).

Tijdens een schuifactie van de gebruiker met de muis (met knop ingedrukt de muis bewegen) is het zaak om bij te houden welk pad de muiscursor aflegt, zodat hieruit de snelheid en richting kan worden afgeleid bij loslaten van de muisknop. Op basis van het opgeslagen pad kan de animatie ingesteld worden om de kaart door te laten schuiven in de richting waarin de gebruiker de kaart bewoog (het eindpunt van de animatie wordt aan de hand van het pad 'geëxtrapoleerd').

Verder kan bij het uitvoeren van de animaties ook een afremmingsfactor worden gebruikt,



(a) Doorschuifsnelheid blijft eerst een tijdje gelijk en neemt daarna geleidelijk af.

(b) Positie blijft constant veranderen (constante snelheid), maar wordt na een tijdje afgeremd.

Figuur 5 - Kaart schuift eerst nog door en komt vervolgens langzaam tot stilstand.

zodat het doorschuiven van de kaart langzamer en langzamer gaat en de kaart langzaam tot stilstand komt (zie figuur 5).

Punt van aandacht bij het gebruik van geanimeerd zoomen en pannen is dat lopende animaties moeten kunnen worden afgebroken. Dus een nieuwe inzoomactie moet bijvoorbeeld de geplande, lopende animatie afbreken (bijvoorbeeld naar aanleiding van een schuifactie), om vanaf het punt dat de gebruiker ingebroken heeft een nieuwe animatie in te plannen.

De eindgebruiker kan in onze implementatie de volgende parameters, behorende bij de geanimeerde zoom- en schuifinteracties, instellen (zie figuur 1(a) onder het getoonde kaartbeeld):

- Grootte van de sprongen die de gebruiker maakt met in-/uitzoomen (hoe groot is de schaalfactor die correspondeert met een klik van het muiswiel, standaard een factor 2).
- De tijdsduur die de animatie neemt nadat de gebruiker klaar is met de interactie, zowel voor in-/uitzoomen als voor het schuiven van de kaart (bijvoorbeeld twee seconden).
- De grootte van de afstand (het na-ijleffect) bij schuiven (dit is in de huidige implementatie gekoppeld aan de snelheid waarmee de gebruiker de kaart geschoven heeft. Dit zou ook zo gemaakt kunnen worden, dat de kaart blijft schuiven voor een wat langere tijd, bijvoorbeeld als de gebruiker de kaart een flinke zwiep geeft).

Door de tijdsduur parameter voor de animatie op 0 seconden te zetten, volgt direct het tonen van het kaartbeeld op het eindpunt en toont het systeem geen animatie van het kaartbeeld. Dit maakt het mogelijk een geleidelijke interactie goed te vergelijken met interactie, die meer stapsgewijs aanvoelt.

Conclusie en verder onderzoek

In dit artikel hebben we laten zien, dat de eerste stappen zijn gezet met een online platform voor het weergeven van vario-schaal data met meer geleidelijke overgangen, gebaseerd op het SSC-model geïmplementeerd in de vorm van een web client. Voor het eerst zijn we hiermee in staat om de resultaten van de vario-schaal aanpak op waarde te schatten en geanimeerd over het web te tonen. Er zijn echter nog wel de nodige uitdagingen en we hebben ook een 'wensenlijstje' voor verdere uitbreiding:

- De geleidelijke overgangen van objecten zijn minder goed waarneembaar dan gehoopt. Daarom is het zaak om de 3D-data die de beschrijving geeft aan te passen, zodat deze overgangen beter zichtbaar worden. Wellicht is dit afhankelijk van objectklassen op de kaart (een andere geleidelijke overgang voor een groepje huizen, dan voor een eenzaam watertje).
- Om goed te kunnen testen, willen we een grotere testdataset beschikbaar maken (bijvoorbeeld heel Nederland). Het gereedschap om een grote tGAP te bouwen is beschikbaar, maar deze omvormen naar een ruimte-schaal kubus, onderverdeeld in blokken, vergt nog de nodige 'engineering'.
- Welke functionaliteit moet waar? Een aantal van de stappen om te komen tot de 3D SSC-data wordt nu als pre-processing stap uitgevoerd, waarbij de blokken met driehoek geometrie uiteindelijk op de server kant worden klaargezet en aangeboden. Dit is echter niet de enige optie. De knip kan ook ergens anders tussen server en client gelegd worden, dat wil zeggen: wie doet wat? Kan de driehoeksdata real time uit blokken data van de tGAP structuur op de client worden afgeleid? Dit heeft pas voordeel als deze data bijvoorbeeld

compacter gecodeerd kunnen worden dan de driehoeksdata.
- Geanimeerd springen van de ene naar de andere plaats op de kaart (bijvoorbeeld door het ingeven van een plaatsnaam), waarbij data van tevoren al kunnen worden overgehaald (en waarbij de kennis over waar de gebruiker binnen x seconden zal gaan kijken meegenomen kan worden). Omdat het pad door ruimte en schaal al kan worden vastgesteld, doordat het start- en eindpunt bij het begin bekend zijn (iets wat bij vrije beweging van gebruikers niet zo is), kunnen de blokken gesorteerd op volgorde aangevraagd worden, wanneer ze nodig zijn.
- Bij vrije beweging van de gebruiker: voorspellen waar de gebruiker naar toe gaat (aan de hand van het al afgelegde pad). Als de voorspelling goed werkt, is het mogelijk om al data van de server aan te vragen, voordat de gebruiker hier kijkt. Hierdoor zal mogelijk de responsiviteit van het systeem toenemen (tenzij de voorspelling niet goed is, dan wordt onnodig data overgestuurd en is het zaak om de dataoverdracht stop te kunnen zetten).
- Verder aanpassen van de client, zodat deze ook werkt op mobiele apparaten, zoals telefoon of tablet, met interactie door een aanraakscherm en bewegingen over zo'n scherm. Nadat dit gebeurd is, willen we de vario-schaal aanpak aan verder gebruikersonderzoek onderwerpen, met behulp van moderne technieken zoals eye-tracking.

Erkentelijk

Dit artikel bevat elementen van het afstudeerwerk van Yueqian Xu, Adrie Rovers en Mattijs Driel. Wij zijn Elmar Eisemann en Timothy Kol zeer erkentelijk voor hun hulp tijdens het begeleiden van deze afstudeerders.

Referenties

- Adrie Rovers, Exploring the Use of a Generic Spatial Access Method for Caching and Efficient Retrieval of Vario-scale Data in a Client-Server Architecture, Master's thesis, Delft University of Technology, pp. 101, 2016.
- Adrie Rovers, Martijn Meijers, Peter van Oosterom, Using a generic spatial access method for caching and efficient retrieval of vario-scale data in a server-client architecture, In: Proceedings of the 20th AGILE Conference on Geographic Information Science (Arnold Bregt, Tapani Sarjakoski, Ron van Lammeren, Frans Rip, eds.), Wageningen University & Research, pp. 6, 2017.
- Yueqian Xu, Construction of a Responsive Web Service for Smooth Rendering of Large SSC Dataset and the Corresponding Preprocessor for Source Data, Master's thesis, Delft University of Technology, pp. 82, 2017.

Martijn Meijers is onderzoeker GIS technologie bij de TU Delft. Hij is bereikbaar via B.M.Meijers@tudelft.nl.

Peter van Oosterom is professor GIS technologie bij de TU Delft. Hij is bereikbaar via P.J.M.vanOosterom@tudelft.nl.

Personalia

Kennismaking Kees Jansen, nieuw GIN-bestuurslid

Kees Jansen (1972) is stadsfilosoof en expert op het gebied van slimme circulaire steden en regio's. Hij is afgestudeerd planoloog en heeft ruime ervaring als adviseur, projectleider, coach en trainer door het hele land. Sinds 2010 heeft Kees zich gespecialiseerd in slimme technologie en de principes van circulair duurzaam denken en handelen in steden. Kees geeft workshops en cursussen op maat en wordt vaak gevraagd voor keynote speeches op evenementen en inspiratiesessies.

Kees is docent slimme circulaire steden aan de Aeres Hogeschool Almere en studieleider van de opleiding Geo Media & Design. Zijn portefeuille in het GIN-bestuur is onderwijs. Kees wil zich het komende jaar richten op een hernieuwde kennismaking met de geo-opleidingen in Nederland en de verbinding van de opleidingen aan het GIN versterken. Met als persoonlijke missie om studenten in aanraking te laten komen met

de sterk veranderende geo-wereld en de schakel te zijn tussen de jonge aankomende en de reeds gevestigde generatie geo-professionals. De geo-wereld mag, wat Kees betreft, wel wat eigentijdser, hipper en sexyer, en een nieuwe generatie brengt een frisse blik mee.

Zijn interesses op het gebied van geo zijn behalve het onderwijs ook de 4D-geo-info (dus 3D plus het aspect tijd) en het metabolisme en de stromen in steden en regio's. Hij ziet de gebouwde omgeving als een organisme waar stromen inkomen, verwerkt worden en het geheel weer verlaten. Daarnaast vraagt hij zich als stadsfilosoof af, wat alle nieuwe manieren van data verzamelen doet met mensen, en waarom we zo veel data verzamelen als we met meer dan 80 procent van de data helemaal niets doen. Met deze drie onderwerpen gaat hij de komende jaren aan de slag.

Kees Jansen is getrouwd en woont in Amsterdam in de Kolenkitbuurt; een dynamische, multiculturele buurt, waar de uitdagingen van het stedelijk leven zich op kleine schaal afspelen. Hij is actief in de buurt en doet veel vrijwilligerswerk. Op deze manier wil hij bijdragen aan het mooier maken van onze wereld.