PRIKAZ VELIKIH KOLIČIN GEOMETRIJSKIH PODATKOV PREKO INTERNETA

Sebastian Krivograd, * izr. prof. dr. Borut Žalik *

Izvleček

V članku podamo zasnovo sistema, kako preko interneta prenesti in prikazati velike količine geometrijskih podatkov, potrebnih za prelet terena v realnem času. Današnji komunikacijski kanali namreč ne dopuščajo prenosa takšnih količin podatkov v enem delu v realnem času, pa tudi prikaz ne poteka brez težav. Reševanje tega problema sestoji iz treh korakov: priprava podatkov (predstavitev podatkov s trikotniško mrežo, poenostavljanje trikotniških mrež, razdelitev podatkov v več modelov različnih natančnosti, stiskanje modelov), posredovanje podatkov s strežnikove strani ter sprejemanje podatkov na strani odjemalca ter prikaz le-teh. z njihovim prikazovanjem. Izkaže se, da v predlagani rešitvi že razmeroma nizka pasovna širina komunikacijskega kanala zadostuje za prenos velikih količin geometrijskih podatkov in prikaz le-teh v realnem času.

KLJUČNE BESEDE: internet, GIS, 3D model, decimiranje, USP

Abstract

In the paper, we propose a scheme for the system able to transmit over the internet and visualise a huge amount of geometric data for making flight through in real time. Namely, today's communication canals with their throughput do not allow transmissions of the whole data set in a real time, and besides this, the visualisation causes some additional problems. The proposed solution consists of three steps: data preparation (presenting the data with a triangular mesh, mesh decimation, data division into several resolution models, compression of these models), data transmission from the server to a client, and receiving the data on the client's side and their visualisation. It is expected that this solution will enable to transmit and visualise huge geometric data sets in a real time.

KEY WORDS: internet, GIS, 3D model, USP

1. UVOD

Že od prvih zametkov računalniške tehnologije je človek poskušal uporabiti računalniški stroj za grafično ponazoritev podatkov, izdelavo skic in modelov. Pri svoji želji pa je naletel na množico težav, ki so izvirale iz dejstva, da so svet in objekti tridimenzionalni in zvezni, računalniški pomnilnik, ki naj bi opis objektov hranil, pa je diskreten in enodimenzionalen. Principi, ki nam povedo, na kakšne načine najučinkoviteje premagati opisani paradoks,

* Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Laboratorij za geometrijsko modeliranje in algoritme multimedije

imenujemo geometrijsko modeliranje [Žalik 1999]. S pojavom in eksplozivnim širjenjem interneta se je zgodba ponovila - razvijalci in uporabniki so zahtevali grafično obogatitev pustega tekstovnega sveta. Opremljeni s tehnikami stiskanja podatkov, so dvodimenzionalne slike kaj hitro postale spremljevalke internetnih aplikacij [Salomon 1997]. Pri 3D modelih pa se je pojavila nova, še neobvladana ovira: pasovna širina komunikacijskega kanala. Klasične predstavitvene metode so se pokazale neučinkovite, saj niso bile prilagojene nizki hitrosti prenosa podatkov.

Prvi resnejši poskusi kako opisati tridimenzionalne geometrijske objekte v okolju internet, je bil standard VRML (Virtual Reality Modeling Language) [VRML 1996]. Na žalost pa so snovalci VRML zaradi prenosljivosti na različne platforme opisali model kar v tekstovni datoteki. S tem je VRML postal primeren le za manjše modele, ki jih najprej uporabnik v celoti prenese, nato pa brskalnik VRML sceno prikaže in uporabniku omogoči premik po njej [Klajnšek 2000]. Ni pa VRML primeren za realne aplikacije z veliko količino geometrijskih podatkov, kot je to v primeru geografskih informacijskih sistemov (GIS). Če za primer vzamemo digitalni model Slovenije z natančnostjo 25 x 25 metrov (DMR25), bi morali prenesti in prikazati okoli 400 MB podatkov. Večina uporabnikov in pa današnjih računalnikov bi se takšnemu početju zagotovo uprli.

V članku bomo prikazali zasnovo sistema strežnik-odjemalec, ki omogoča vizualizacijo velikih količin geometrijskih podatkov, kot primer uporabimo digitalni model reliefa, organiziranih v trikotniške mreže preko interneta. Ob zagotovitvi minimalne pasovne širine in minimalne konfiguracije odjemalca sistem omogoča vizualizacijo modela terena v realnem času.

Članek je organiziran v štiri poglavja. V drugem poglavju podamo osnovno konfiguracijo sistema, v tretjem bomo predstavili najzahtevnejši del priprave podatkov - poenostavljanje trikotniške mreže in v zadnjem bomo opisali minimalne zahteve odjemalca.

2. KONFIGURACIJA SISTEMA

Ključne zahteve in omejitve, ki naj jim zadosti sistem za prenos in vizualizacijo velikih količin podatkov v okolju internet (VisNet3D), so:

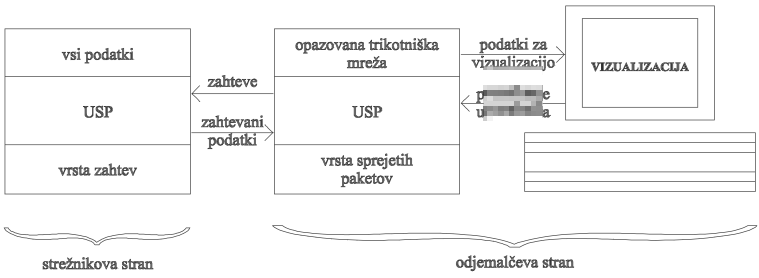
- želimo internet aplikacijo, temelječo na arhitekturi strežnik-odjemalec,
- odjemalci so lahko priključeni na internet z različno prepustnostjo komunikacijskih kanalov,
- konfiguracije odjemalcev so lahko različne, zadostiti pa morajo nekaterim minimalnim zahtevam, opisanim v zaključku tega članka,

- proces vizualizacije opravimo v strojni opremi preko standarda OpenGL [OpenGL 1999],
- strežnik mora na podlagi informacije o položaju opazovalca dostaviti podatke, potencialno vidne s strani opazovalca,
- strežnik lahko streže končnemu številu odjemalcev.

Da bi omenjenim zahtevam lahko zadovoljili, uporabimo naslednje tehnike:

- podatke DMR organiziramo v enakomerno mrežo (Uniform Space Partition - USP), sestavljeno iz množice celic,
- v vsaki celici tvorimo triangulacijo iz originalnih točk DMR (trikotnike najhitreje vizualiziramo s strojno opremo),
- s postopkom poenostavitve trikotniških mrež (triangulation declination) pripravimo trikotniške mreže različnih ločljivosti, ki jih hranimo v podatkovni bazi na strani strežnika,
- topologijo trikotniških mrež in koordinate točk stisnemo in s tem zmanjšamo zahtevano hitrost prenosa.

Osnovno konfiguracijo sistema prikazuje Slika 1. Odjemalec vzpostavi USP, ki ustreza podatkom v podatkovni bazi. Glede na položaj opazovalca pošlje zahtevo strežniku po podatkih. Strežnik najprej pošlje stisnjene podatke z najmanjšo ločljivostjo trikotnikov. Če se uporabnik premika nad terenom dovolj počasi, bo imel strežnik dovolj časa za prikaz vedno večjih podrobnosti, zato mu strežnik pošlje trikotnike z naslednjega nivoja. Kakor hitro se uporabnik približa robu celice, preveri, ali podatkov o celici še nima v svojem vmesnem pomnilniku. Če jih nima, pošlje zahtevo strežniku, ki mu pošlje paket trikotnikov z najmanjšo resolucijo.



Slika 1: Struktura celotnega sistema

Na kratko si še oglejmo idejo implementacije strežnika in odjemalca.

- Strežnik posreduje podatke iz podatkovne baze (v našem primeru je organiziran kot preprost datotečni sistem) na podlagi informacije o položaju in nivoju. Glede na to, da se trenutno lahko pojavi več zahtev, uporabimo čakajočo vrsto zahtev. Za realizacijo uporabimo dve niti:
  - Nit za sprejemanje zahtev od odjemalcev:

```
while (TRUE)
{  sprejmi zahtevo od odjemalca
   shrani zahtevo v vrsto zahtev
}
```

  - Nit za pošiljanje podatkov glede na zahteve:

```
while (TRUE)
{  if kakšna zahteva v vrsti zahtev
   {  vzemi zahtevo iz vrste zahtev
      pošlji podatke odjemalcu, ki je poslal zahtevo
   }
}
```

- Implementacija odjemalca je nekoliko zahtevnejša, saj mora nadzorovati naslednje postopke:
  - Zahteva po podatkih. Tu ločimo med začetno zahtevo, preden začnemo s preletom ter zahtevo po podatkih, ki izboljšajo resolucijo trikotniške mreže.
  - Posredovanje podatkov strojni opremi za vizualizacijo.
  - Brisanje podatkov, ki niso več potrebni na odjemalčevi strani.

Vse te operacije tečejo v treh nitih:

Glavna nit

```
pošlji zahtevo po osnovnih podatkih glede na
uporabnikovo začetno pozicijo in čakaj na
odgovor
while (TRUE)
{  glede na uporabnikovo pozicijo vstavi
     pakete iz vrste sprejetih paketov v
      trenutno trikotniško mrežo
   if (trenutna trikotniška mreža > MAXSIZE)
      odstrani nepotrebne pakete iz
        opazovalne trikotniške mreže
   vizualiziraj opazovalno trikotniško mrežo
}
```

- Nit za pošiljanje zahtev

```
while (TRUE)
{  najdi smerni vektor uporabnikovega
   premikanja
   potuj v smeri tega vektorja in najdi
   manjkajoče pakete
   pošlji zahtevo po manjkajočih paketih
}
```

- Nit za sprejemanje zahtevanih podatkov

```
while (TRUE)
{  if (sprejet je paket od strežnika)
   {  vstavi paket v vrsto sprejetih paketov
      if (dolžina vrste sprejetih paketov >
          MAXSIZE)
                odstrani najmanj potreben
                      paket iz vrste
   }
}
```


3. POENOSTAVLJANJE TRIKOTNIŠKE MREŽE

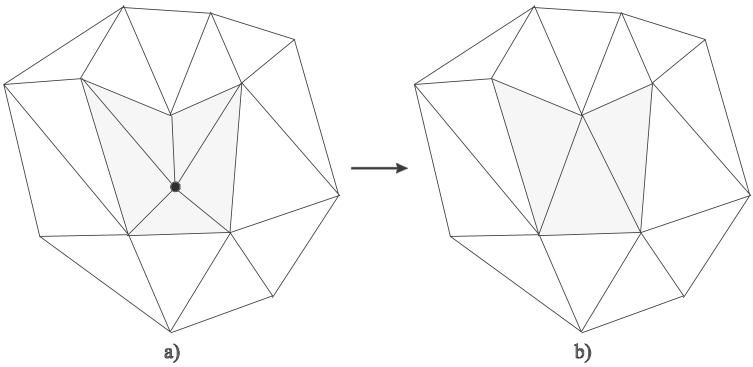
Za hitrejše delo moramo podatke pripraviti vnaprej. Pri tem se izvedejo naslednja opravila:

- predstavitev podatkov z enakomerno trikotniško mrežo,
- poenostavljanje trikotniške mreže,
- razdelitev podatkov v trikotniške modele na več nivojih natančnosti,
- stiskanje trikotniških modelov.

Najzahtevnejše opravilo je poenostavljanje trikotniške mreže, zato bomo idejo na kratko opisali. S poenostavljanjem trikotniške mreže želimo zmanjšati število točk in trikotnikov, ki predstavljajo relief, s čim manjšim odstopanjem od originalnega reliefa. Preden začnemo odstranjevati točke, jih moramo nekako ovrednotiti, da vemo, pri katerih je odstopanje najmanjše po njihovi odstranitvi. Pri DMR vrednotenje temelji na povprečni višinski razliki obravnavane točke do sosednjih točk (do točk, s katerimi tvori trikotnike). Ko izračunamo faktor ovrednotenja vsem točkam, začnemo z odstranjevanjem točk. Točke razdelimo v n razredov glede na normiran

faktor ovrednotenja. Zatem lahko pričnemo s poenostavljanjem (decimiranjem) trikotniške mreže. Vzamemo prvo točko v najnižjem nepraznem razredu. To točko nato odstranimo iz razreda ter iz trikotniške mreže skupaj z vsemi trikotniki, ki jih ta točka določa (temnejši trikotniki na Sliki 2a). V triangulaciji zato nastane luknja, ki jo moramo zapolniti z novimi trikotniki (Slika 2b). S tem pa se spremeni faktor ovrednotenja vsem neposredno sosednjim točkam odstranjene točke. Te točke najprej odstranimo iz razredov, nato jih ponovno ovrednotimo in jih nazadnje vstavimo na konec ustreznih razredov. S prestavitvijo ponovno ovrednotenih točk na konec razredov smo dosegli, da poenostavljanja trikotniške mreže ne opravljamo samo na enem mestu (lokalno) ampak enakomerno po celotnem reliefu.

Slika 2: Odstranitev točke



4. ZAKLJUČEK

V članku podamo zasnovo sistema, ki omogoča dinamično vizualizacijo velikih trikotniških mrež preko interneta. Najprej opišemo zasnovo sistema, nato podamo osnovne funkcije, ki jih morata opraviti strežnik in odjemalec. Sistem je trenutno še v fazi implementacije, zato podajamo samo ocene zahtev za strojno opremo odjemalca in pasovno širino komunikacijskega kanala. Odjemalec naj ima procesor Pentium II in 128 MB hitrega pomnilnika. Grafična kartica mora imeti strojno podporo OpenGL z vsaj 32 MB pomnilnika. Glede na dosedanje izkušnje velja, da je za realno vizualizacijo potrebno prikazovati vsaj 4000 trikotnikov [Floariani 2000], ki so opisani s približno 2000 točkami. Za vsako točko potrebujemo 11 B, če njihove koordinate x, y predstavimo kot 32 bitna cela števila, nadmorsko višino pa s 24 biti. Za opis točk trikotniške mreže tako potrebujemo 22KB, za opis topologije trikotniške mreže pa še dodaten 1KB [Gumhold 1998]. Skupaj torej potrebujemo 23KB, ki jih z dodatnim algoritmom stiskanja stisnemo vsaj za 80%. Vseh 4000 trikotnikov, potrebnih za vizualizacijo, se ne spremeni v eni sekundi, zato je zgornja ocena za hitrost komunikacijskega kanala 4,6 KB/s, kar glede na današnjo tehnologijo ne predstavlja ovir.

Literatura

***L. De Floriani, P. Magillo, F. Morando, E. Puppo**, »Dynamic view-dependent multiresolution on a client-server architecture«, Computer-Aided Design, Vol. 32, 2000, pp. 805 – 823 [Floariani 2000]*

***Stefan Gumhold, Wolfgang Straßer**, »Real Time Compression of Triangle Mesh Connectivity«, SIGGRAPH '98, Proceedings of the 25th annual conference on Computer Graphics, July 19 - 24, 1998, Orlando, FL USA, pp.133 – 140 [Gumhold 1998]*

***Klajnšek, G., Zadravec, M., Podgorelec, D., Žalik, B.**, »Representing the sights of a town using VRML«. Proceedings of 22nd International conference on Information technology interfaces, Pula, Croatia, June 13-16, 2000, pp. 235-240. [Klajnšek 2000]*

***»OpenGL Programing Guide**: The Official Guide to Learning OpenGL, Version 1.2«, Addison Wesley Longman, Massachusetts, 1999 [OpenGL 1999]*

***David Salomon**, »Data Compression: The Complete Reference«, Springer Verlag, New York, 1997 [Salomon 1997]*

***John R. Vacca**, »VRML: Bringing Virtual Reality to the Internet«, Academic Press Limited, London, 1996 [VRML 1996]*

***Borut Žalik**, »Geometrijsko Modeliranje«, Fakulteta za elektrotehniko, računalništvo in informatiko, 1999 [Žalik 1999]*