Geometrija (smer Informatika)
Seminarski rad

Srdjan Vukmirovic

December 3, 2008

Opšta uputstva za izradu seminarskog

- Seminarski radi jedan student ili dva studenta u grupi.
- Kada odaberete seminarski pošaljite e-mail na tijana@matf.bg.ac.yu. U e-mailu navedite koji ste seminarski odabrali i da li radite sami. Ako dva studenta rade u grupi potrebno je da oba pošalju e-mail.
- Nije neophodno da se student strogo pridržava date specifikacije. Dozvoljena je svaka promena imena, tipova funkcija, argumenata i načina rada koje vam se učini korisnom.
- Upoteba (jednostruko ili dvostruko povezanih) lista umesto nizova je pozeljna, gde god to ima vise smisla.
- Programe je potrebno pisati jasno, koristeći komentare. Kada se pišu pomoćne funkcije potrebno je u komentarima kratko objasniti čemu one služe.
- Za osnovne objekte tačku, pravu, duž i krug treba koristiti strukture iz skripti.
- Svaki seminarski treba da ima graficki prikaz. Dovoljno je da se taj prikaz uradi uz pomoć funkcije `nacrtaj`, koja ce biti uskoro modifikovana da omogući laksi rad. Ko želi, moze da uradi svoj grafički prikaz (vodite računa, ovo ce vam oduzeti vremena, osim ako znate kako to da uradite).
- Krive (Bezijerove, elipse, hiperbole, parabole...) se crtaju tako što se aproksimiraju dužima, koje se dobijaju iz parametrizacije krive.

1 Elementarne Bezijerove krive

a) Treba napisati funkciju

`int Bezijer2 (Tacka t1, Tacka t2, Tacka t3, float nizx, float nizy)`.

Argumenti funkcije su tri kontrolne tačke Bezijerove krive drugog reda. Funkcija treba da vrati 0 ako Bezijerova kriva ne postoji (tj. te tri tačke su kolinearne), a 1 ako ta kriva postoji. Funkcija treba da upiše u nizove `nizx` i `nizy` koeficiente bezijerovih polinoma stepena 2 za x i y koordinatu krive. Nizovi su dužine 3. Recimo, ako je $t1 = (1, 7)$, $t2 = (2, -3)$, $t3 = (7, 9)$ tada je Bezijerova kriva

$$\alpha(t) = t^2(1, 7) + 2t(2, -3) + (1 - t)^2(7, 9) = (7 - 10t + 8t^2, 9 - 24t + 16t^2)$$

nizovi bi trebalo da budu $nizx = \{7, -10, 8\}$ i $nizy = \{9, -24, 16\}$.
b) Treba napisati funkciju

`int Bezijer3 (Tacka t1, Tacka t2, Tacka t3, Tacka t4, float nizx, float nizy)`.

Argumenti funkcije su četiri kontrolne tačke. Funkcija treba da vrati 0 ako Bezijerova kriva ne postoji (tj. tačke `t1, t2, t4` ili `t1, t3, t4` su kolinearne), a 1 ako ta kriva postoji. Nizovi `nizx, nizy` su dužine 4 i funkcija u njih treba da upiše koeficiente polinoma stepena 3 slično kao pod a).
c) Napisati programe koji pozivaju funkcije pod a), odnosno b). Korisnik treba da unese, tri, odnosno četiri tačke, a program treba da na ekranu ispiše krivu u obliku, recimo, `a(t) = (7 - 10 t + 8 t^2 , 9 - 24 t + 16 t^2)` i da nacrta krivu, kontrolne tačke i odgovarajuće tangente.

2 Bezijerova kriva kroz n tačaka

Ovo je jednostavan zadatak, za koji je potrebno malo matematičkog razmišljanja.
a) Treba napisati funkciju

`int BezijerN (Tacka tniz, int n, float nizx, float nizy)`.

Argumenti funkcije su niz tačaka `tniz` dužine `n+1`> 2 i nizovi realnih brojeva `nizx` i `nizy`. Nizovi `nizx` i `nizy` su dužine $3(n - 2)$.

Treba napraviti niz od $n-2$ Bezijerove krive stepena 2, tako da su kontrolne tačke prve krive, redom, `tniz[1], tniz[0]` i `tniz[2]`. Kontrolne tačke druge su `tniz[2],` M_1 i `tniz[3]`, gde je tačka M_1 simetrična sa `tniz[0]` u odnosu na `tniz[2]`. Kontrolne tačke treće krive su `tniz[3],` M_2 i `tniz[4]`, gde je M_2 simetrična sa M_2 u odnosu na `tniz[3]`, itd.

Koeficiente polinoma tih $n-2$ Bezierovih krivih treba upisati u `nizx` i `nizy` (vidi zadatak Elementarne Bezijerove krive a)), tako da prva tri elementa sadrže koeficiente prve krive, druga tri, druge krive itd...

Funkcija treba da vrati 0 ako su prve tri tačke kolinearne (tada sve nema puno smisla), a 1 inače.
b) Napisati program koji testira funkciju pod a) tako što od korisnika traži da izabere da li će tačke unositi preko tastature ili iz fajla. Ako se tačke unose sa tastature treba na ekranu ispisati $n-2$ krivih u obliku, recimo,

```
a01(t) = (7 - 10 t + 8 t^2 , 9 - 24 t + 16 t^2)
a02(t) = (5 - 11.45 t + 8.23 t^2 , 1.9 + 2.4 t + 1.3 t^2)
....
```

a ako se unose iz fajla, tada isto to treba upisati u fajl.
c) Nacrtati kontrolne tačke i i Bezijerove krive. Trebalo bi da se (zbog simetrije) svake dve susedne Bezijerove krive glatko spajaju zajedničkoj kontroloj tački. Testirati šta se dešava kada se menja samo tačka `tniz[0]`.

O poligonima

U funkcijama koje se odnose na poligone podrazumevamo da je poligon $P_0P_1 \dots P_n$ zatvoren, tj. $P_0 = P_n$. Ko koristi nizove, poligon pamti kao niz tačaka. Ko koristi liste, poligon smatra listom ili dvostruko povezanom listom, šta smatra zgodnijim. Onaj ko koristi liste ne treba da prenosi dužinu poligona kao argument funkcije.

Većina funkcija vezanih za poligone koristi funkciju za presek dve duži ili poluprave i duži, tako da je prvo njih potrebno implementirati.

3 Da li je tačka u unutašnjosti poligona

a) Napisati funkciju

`int Prost (Tacka pol, int n)`.

koja vraća 1 ako je poligon `pol` sa `n` temena prost, a 0 ako poligon nije prost.
b) Napisati funkciju

`int Unutra (Tacka A, Tacka pol, int n)`.

koja vraća 1 ako je tačka A u unutrašnjosti poligona `pol`, -1 ako je u njegovoj spoljašnjosti, a 0 ako pripada poligonu (tj. nalazi se na nekoj od ivica). Broj `n` je broj temena poligona.
c) Napisati program koji testira funkcije pod a) i b) i crta poligon, tačku, polupravu i presečne tačke. Korisnik treba preko tastature (ili iz fajla) da unese temena poligona i tačku A. Ako poslednje uneto teme poligona nije isto kao prvo, treba dodati još jedno teme koje je jednko prvom (tj. treba zatvoriti poligon). Program treba da ispiše na ekran da li je poligon prost i kakav je položaj tačke A u odnosu na poligon.

4 Triangulacija poligona

a) Napisati funkciju

`int Prost (Tacka pol, int n)`

koja vraća 1 ako je poligon `pol` sa `n` temena prost, a 0 ako poligon nije prost.
b) Napisati funkciju

`int UnutrasnjaDijagonala (Tacka pol, int n)`

koja vraća broj k tako da je P_kP_{k+1} unutrašnja dijagonalu poligona `pol`. Podrazumeva se da je poligon prost.
c) Napisati rekurzivnu funkciju

`int Triangulisi (Tacka pol, int n, Trougao tlista)`.

Funkcija treba da poligon `pol` unutrašnjim dijagonalama razbije (trianguliše) na niz trouglova `tniz`, a da vrati broj trouglova u triangualciji (koji mora biti za dva manji od broja temena poligona - provera!) Dakle u listu `tlista` treba upisati trouglove dobijene triangulacijom poligona `pol`. Funkcija treba da poziva funkciju `UnutrasnjaDijagonala` .
d) Napisati program koji testira funkcije pod a) i c) i crta sve. Korisnik treba preko tastature da unese temena poligona. Ako poslednje uneto teme poligona nije isto kao prvo, treba dodati još jedno teme koje je jednko prvom (tj. treba zatvoriti poligon). Program treba da ispiše na ekran da li je poligon prost. Ako jeste, program treba i da pozove funkciju za triagulaciju i ispiše listu trouglova dobijenu triangulacijom datog poligona. Ako nije, tringulaciju ne treba vršiti.

Napomene: `Trougao` je struktura koja se sastoji od 3 tačke, tipa `Tacka`.

Za argument `tlista` ne mora se koristiti lista, ako možete nizom da uradite istu stvar.

5 Konveksni omotac reda (spori algoritam)

a) Napisati funkciju

`void KonveksniOmotacN3 (Tacka tniz, int n, Tacka omotac)`.

koja za niz tačaka `tniz` dužine n računa konveksni omotač i upisuje ga u niz tačaka `omotac`. Funkcija treba da koristi algoritam sa predavanja.

b) Napisati program koji testira funkciju pod a) i crta. Korisnik unosi niz tačaka preko tastature (ili iz fajla), a program treba da ispiše (na ekran ili u fajl) redom niz tačaka koje čine konveksni omotač.

6 Konveksni omotac reda (brzi algoritam - odredjivanjem gornjeg i donjeg omotača)

Isto kao i prethodno.

7 Svodjenje krive drugog reda na kanonski oblik

Napisati program u koji se unosi niz od 6 koeficenata a_{ij} krive drugog reda, a program treba krivu drugog reda da svede na kanonski oblik. Dakle on treba da ispiše: kanonski oblik krive je, recimo `x^ 2/(2.34^2) + y^2/(1.14^2)=1`, da kaže o kojoj krivoj se radi i da napiše formule transformacija.

Pored toga program treba da nacrta originalni koordinatni sistem, zarotirani koordinatni sistem i translirani sistem, kao i krivu. Za crtanje elipse, hiperbole i parabole koristiti njihove parametrizacije.

8 Orjentacija i rub poliedarske površi

a) Napisati funkciju

`int Orjentisi (Pljosan pniz, int n)`.

koja orjentiše pljosni zadate poliedarske površi. Poliedarska površ je zadata nizom (listom) od `n` pljosni `pniz`. Dakle, treba izvršiti uskladjivanje orjentacija pljosni na osnovu orjentacije prve pljosni `pniz[0]`. Preorjentisane pljosni treba upisati u isti niz `pniz`. Funkcija treba da vrati 1 ako je površ orjentabilna (tj. uskladjivanje ojentacija je moguće), a 0, ako je površ neorjentabilna.

(Svaka pljosan je zadata nizom (listom) indeksa temena. Indeksi temena su prirodni brojevi. Pljosni mogu da imaju razliit broj temena. Koordinate temena isu važne, tj. ne koriste se.)

b) Napisati funkciju

`int Rub (Pljosan pniz, int n)`.

koja odredjuje rub i broj komponenata ruba poliedarske površi.

c) Napisati program koji testira funkcije pod a) i b) **(nema crtanja!)**. Korisnik unosi niz pljosni preko tastature (ili iz fajla), a program treba da ispiše uskladjene orjentacije pljosni ili poruku da je površ neorjentabilna. Takodje treba da ispiše rub i broj komponenata ruba.

9 Projekcije Platonovih tela

a) Napisati funkciju

`int PraviTemena (FILE pfile, Tacka temena)`

koja za dati pokazivač fajla `pfile`, cita koordinate temena platonovog tela, i od njih pravi niz temena `temena` tipa `Tacka`. Naravno, svako teme tj. `Tacka` je opisana sa tri koordinate tipa `float`. Funkcija treba da vrati 1 ako je sve prošlo u redu, inače 0.

b) Napisati funkciju

```
int PraviPljosni (FILE pfile, Pljosan povrs)
```

koja za dati pokazivač fajla `pfile`, čita iz fajla indekse temena koji čine pljosni Platonovog tela i od njih pravi niz struktura `povrs`, tipa `Pljosan`. (Svaka pljosan je zadata nizom indeksa temena. Indeksi temena su prirodni brojevi. Pljosni imaju isti broj temena, 3, 4 ili 5). Funkcija treba da vrati 1 ako je sve prošlo u redu, inače 0.

Fajlovi za funkcije pod a) i b) se mogu skinuti sa http://alas.matf.bg.ac.yu/ vsrdjan/files/geometrija.htm.

c) Napisati program kome korisnik zadaje Platonovo telo (kocka, oktaedar, tetraedar, dodekaedar, ikosaedar). Program zatim koristeći funkcije pod a) i b) iz odgovarajućeg fajla čita podatke o datom telu i prikazuje normalnu i centralnu projekciju (izbor korisnika) tela na ekranu.

Prikaz nevidljivih ivica isprekidanim linijama nije obavezan, mada bi ga bilo interesantno videti.