Laboratorijske vaje pri predmetu Osnove in Postopki Konstruiranja (OPK)

Janez Krek, Laboratorij za CAD - LECAD, šolsko leto: 1999/2000

14. oktober 1999

1 Naloga

Izdelajte program za prikaz 3-D objektov na ekranu v poljubnem pogledu. Prikaz objekta na ekranu z žičnim modelom izvedite s homogenimi transformacijami poljubnega zaporedja. Program naj omogoča zapis objekta v dve VRML datoteki: kot žični model in kot telo. Za program izdelajte poročilo v HTML obliki z vključenimi primeri uporabe programa ter prikazom objekta, zapisanega v VRML obliki.

2 Izdelava programa

Programskih jezikov, v katerem bi lahko napisali program za izvedbo naloge, je zelo veliko. Ker je zahteva programa tudi izris grafike, dajemo na izbiro naslednje možnosti:

- JavaScript + Phigs
- Java
- C + Phigs

3 Vhodni podatki

Programu morate podati objekt, ki ga nariše na ekranu. To naredite s pomočjo vhodne datoteke (če pišete program v **C**ju) ali pa vnesete podate preko tipkovnice (pogramiranje v **Javi** ali **JavaScriptu**). Pri programu, napisanem v **Javi** ali **JavaScriptu** vnos podatkov naredite preko vnosnega polja za vnos teksta (neke vrste multi-line edit), v katerega lahko vhodne podatke vnesete preko odložišča (clipporda).
Vhodni podatki vsebujejo:

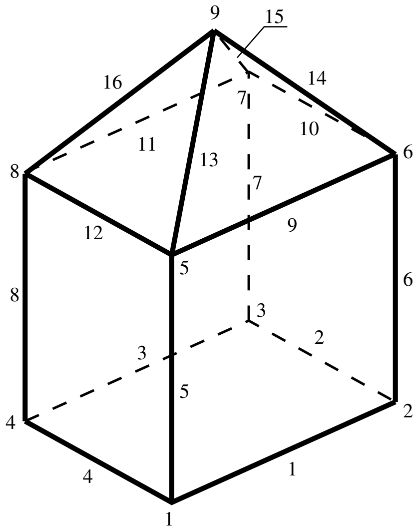
- podatke o geometriji objekta:
  - točke
  - robove
  - ploskve
- podatke o barvah
- začetni položaj pogleda

3.1 Primer vhodne datoteke

Vhodna datoteka je ASCII datoteka, ki ima predpisano obliko podatkov in jo lahko napišete s poljubnim ASCII urejevalnikom. Primer vhodne datoteke za objekt na sliki 3.1:

```
# Primer vhodne datoteke
ime Primer 1

# najprej so dolocene koordinate tock
# oznaka za tocke in stevilo tock
# vsaka vrsta ima format: st. tocke x y z
tocke 9
1 0 0 0
2 3 0 0
3 3 2 0
4 0 2 0
5 0 0 3
6 3 0 3
7 3 2 3
8 0 2 3
9 1.5 1 5

# nato dolocimo robove (za zicni model)
# vsaka vrsta ima format: st. roba T1 T2
robovi 16
1 1 2
2 2 3
3 3 4
4 4 1
5 1 5
6 2 6
7 3 7
8 4 8
9 5 6
10 6 7
11 7 8
12 8 5
13 5 9
14 6 9
15 7 9
16 8 9

# in na koncu podamo se ploskve (za sencenje)
# vsaka vrsta ima format:
#   st. ploskve barva st. robov R1 R2 R3 R4 .... Rn
ploskve 9
1 2 4 1 2 3 4
2 2 4 1 6 9 5
3 2 4 2 6 10 7
4 2 4 3 8 11 7
5 2 4 4 5 12 7
6 4 3 9 14 13
7 4 3 10 15 14
8 4 3 11 16 15
9 4 12 13 16

# barve
# podatki: st.barve R G B
barve 8
1 0 0 0
2 255 0 0
3 0 255 0
4 0 0 255
5 255 255 0
6 255 0 255
7 0 255 255
8 255 255 255
```

```
# zacetni polozaj pogleda
# podatki: x y z
pogled -10 -10 7
```



Vhodna datoteka je razdeljena na 6 delov: ime objekta, tocke, robovi, ploskve, barve in pogled. Na začetku vsakega odseka je njegovo ime in število elementov v odseku (razen pri *ime* in *pogled* kjer ni elementov). Nato so elementi posameznega odseka napisani po eden v vsaki vrsti. Prazne vrstice in vrstice, ki se začnejo z znakom $\#$, se pri branju zanemarijo (ne upoštevajo).

3.2 Branje vhodnih podatkov

Ne glede na to, kako se podaja vhodne podatke, je le-te potrebno prebrati in zapisati v obliki, da je manipulacija z njimi čimlažja. Podatki so razdeljeni na 6 delov in vsak del se bere posebej. Najbolj učinkovit način shranjevanja podatkov (točk, robov in ploskev) je s tabelami oziroma polji.

4 Prikaz objekta na ekranu

Prvi del programa je prikaz objekta na ekranu. Program mora omogočati poljuben pogled na objekt. Objekt na ekranu prikažite kot žični model (objekt je prikazan s svojimi robovi).

4.1 Homogene koordinate

Uporaba homogenih koordinat poenoti zapis geometrijski transformacij (translacija, rotacija in skaliranje). S tem je manipulacija s točkami enostavnejša. Primerjavo zapisov točk prikazuje tabela 1.

kartezijske	homogene
T[x,y,z]	T[x,y,z,1]

Tabela 1: Zapis točke v homogenih koordinatah

4.2 Geometrijske transformacije

Objekt na ekranu lahko skalirate, translirate ali rotirate. Z zapisom točk v homogenih koordinatah se te transformacije zapišejo z eno enačbo (enačba 1):

$$\vec{A}_2 = [T] \cdot \vec{A}_1^T \tag{1}$$

ali v razširjeni obliki z enačbo 2

$$[x, y, z, 1]_2 = [T] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_1 \tag{2}$$

kjer oznake pomenijo:

- $\vec{A}_2$ - končna (izračunana) točka
- T - transformacijska matrika
- $\vec{A}_1$ - začetna (izvorna) točka

Matrika T je odvisna od transformacije in je naslednje oblike:

- **rotacija**: matrika za rotacijo okoli osi za podan kot ϕ:
  - X: $[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\phi & sin\phi & 0 \\ 0 & -sin\phi & cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
  - Y: $[T] = \begin{bmatrix} cos\phi & 0 & -sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ sin\phi & 0 & cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
  - Z: $[T] = \begin{bmatrix} cos\phi & sin\phi & 0 & 0 \\ -sin\phi & cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- **translacija**: matrika za translacijo za Δx, Δy in Δz:

$$[T] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \Delta x & \Delta y & \Delta z & 1 \end{bmatrix}$$

- **skaliranje**: povečava/pomanjšanje za faktor S:

$$[T] = \begin{bmatrix} S & 0 & 0 & 0 \\ 0 & S & 0 & 0 \\ 0 & 0 & S & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.3 Risanje na ekran

Pred izrisom objekta na ekranu, je potrebno objekt transformirati v skladu z vnesenimi transformacijami. Najprej transformirate vse točke objekta in nato narišete vse robove, ki sestavljajo objekt. Kako narisati točko v 2-D, če jo imamo 3 koordinate? Pri risanju točk upoštevate upoštevate le X in Y koordinato.

5 Zapis v VRML

Drug del naloge je zapis objekta v VRML datoteki: eno kot žični model (zapis z robovi) in druga kot telo (zapis s ploskvami). Datoteki sta ASCII datoteki.

VRML je kratica za Virtual Reality Modeling Language in predstavlja jezik za popis navidezne resničnosti. Jezik omogoča enostaven zapis 3-D objektov in raznih scen (postavitev objektov v prostor, potovanj med objekti, itd.). Datoteka je sestavljena iz blokov, ki so zajeti znotraj zavitih oklepajev ({ in }). Datoteki, ki jo mora napisati program sta enostavni, saj vsebujeta le podatke o točkah in robovih oziroma črtah (5.1).

Več si lahko o VRML jeziku preberete na internetni strani *http://www.slip.net/∼fxlab/vrmlindex.html* ali na drugin straneh o VRML jeziku.

5.1 Primer datoteke VRML

VRML datoteka objekta na sliki 3.1 (stran 2).

```
#VRML V1.0 ascii
Coordinate3 {
        point[  0 0 0,
                3 0 0,
                3 2 0,
                0 2 0,
                0 0 3,
                3 0 3,
                3 2 3,
                0 2 3,
                1.5 1 5 ]
             }
Material{
        diffuseColor 1 0 0
        }

IndexedFaceSet {
        coordIndex[     0, 1, 5, 4, -1,
                        1, 2, 6, 5, -1,
                        2, 3, 7, 6, -1,
                        3, 0, 4, 7, -1,
                        0, 1, 2, 3, -1 ]
             }
Material{
        diffuseColor 0 1 0
        }

IndexedFaceSet {
        coordIndex[     4, 5, 8, -1,
                        5, 6, 8, -1,
                        6, 7, 8, -1,
                        7, 4, 8, -1 ]
             }
```

5.1.1 Določitev koordinat točk

Koordinate točk se določi na začetku datoteke, ker se indeksi točk uporabijo pri določitvi robov oziroma ploskev objekta.

```
Coordinate3 {
        point[x y z,
              x y z,
              x y z]
           }
```

Točke imajo koordinate medsebojno ločene s presledki. Točke so medsebojno ločene z vejico (,). Zaporedje točk določa indekse točk. Prva točka ima indeks 0, naslednja 1, itd. Z novo definicijo točk, se podatki o starih završejo in definirajo se novi indeksi točk.

5.1.2 Določitev robov

Robovi se določijo z določitvijo točk.

```
IndexLineSet{
        coordIndex [2, 1, 3 -1
                    2, 4, 2 -1]
            }
```

S **coordIndex** se določi točke, ki tvorijo lomljenko (polyline). Z **-1** se označi konec črte (lomljenke). Številke v oklepaju pomenijo indekse trenutno definiranih točk.

5.1.3 Določitev ploskev

Ploskve se določijo z določitvijo točk.

```
IndexedFaceSet{
        coordIndex [2, 1, 3 -1,
                    2, 4, 2 -1]
            }
```

S **coordIndex** se določi točke, ki tvorijo ploskev. Z **-1** se označi konec točk za eno ploskev. Številke v oklepaju pomenijo indekse trenutno definiranih točk.

5.1.4 Določitev barve

Ploskve in robovi se izrisujejo v trenutno določeni barvi, ki se določi z ukazom:

```
Material{
        diffuseColor R G B
        }
```

Vrednosti **R**, **G** in **B** so vsebnost rdeče, zelene in modre barve v želeni barvi. Vrednosti so lahko med 0 in 1.

6 Poročilo

Na koncu vaj, skupaj s programom, oddate tudi poročilo, napisano v HTML obliki. Poročilo mora vsebovati:

- opis naloge
- matematične osnove, na katerih temelji rešitev naloge
- izvedba rešitve (postopki, krajši odseki programske kode), vendar NE celotnega listinga programa!
- zaključek (težave pri programiranju, izboljšave programa, itd)
- primeri, s katerimi je bil program testiran (vhodne podatke, slike v VRML)

Poročilo naj vsebuje kazalo in v primeru velikih slik, v besedilo vstavite manjše slike, ki imajo povezavo na veliko sliko. V poročilu morajo biti predstavljeni primeri objektov, ki so bili uporabljeni pri testiranju programa v VRML obliki.

6.1 Kaj je HTML in kako napišem poročilo

HyperText Markup Language (HTML) je jezik za opis strani, ki se uporablja v internetu. Poročilo lahko napišete na več načinov:

1. poročilo napišete v ASCII urejevalniku (Emacs, Notepad)
2. uporabite HTML urejevalnik (Netscape Composer, Microsoft FrontPage, itd)
3. napišete poročilo v vašen najljubšem urejevlniku besedil (MS Word, WordPerfect, itd) in poročilo na koncu urejanja, zapišete v HTML obliki.

Priporočamo vam, da poročilo napišete na prvi način, ker je najbolj enostaven. Vse kar morate vedeti je nekaj ukazov HTML in poročilo je napisano (seveda, če vsebina ni problem). Tako se skoncentrirate na vsebino poročila in ne na njegovo obliko (kot pri načinu 2 in 3). Najbolj pogosti ukazi HTML jezika so zbrani v dokumentu na straneh LECAD laboratorija (*http://www.lecad.uni-lj.si/documents/navodila/html/navhtml.html*). Pri pisanju poročila pazite na imena datotek (pri povezavah - predvsem slikah). Imena datotek na disku in v HTML dokumentu morajo biti enaka - upoštevajte velike in male črke!

Za pisanje poročila uporabite pisavo **UTF-8**. Spodnji odsek prikazuje, kako v HTML dokumentu nastavite prevzeto pisavo.

```
<HEAD>
<TITLE>Pisanje porocil v jeziku HTML</TITLE>
<META HTTP-EQUIV="Content-Type"
      CONTENT="text/html; charset=utf-8">
</HEAD>
```

Dokument, ki ga lahko uporabite za osnovo pri pisanju poročila, se nahaja na LECAD strežniku (**http://www.lecad.uni-lj.si/vaje/html/**).

6.2 Kako prikažem VRML v HTML dokumentu?

V poročilo morate vstaviti ukaze, ki omogočajo prikaz VRML objekta v HTML dokumentu. VRML datoteko pripravite s programom (prva datoteka, ki vsebuje žični model objekta). Ukazi za prikaz objekta so:

```
<applet code=Phigs.class
    archive="http://www.lecad.uni-lj.si/
             ~vrhovec/vaje/java3d/viewer.jar"
    width=600 height=600>
<param name=bgcol value=black>
<param name=file  value=XXXXXX>
<param name=coord value=false>
</applet>
```

Namesto *XXXXXX* vstavite ime VRML datoteke. Ukaze vnesete v HTML dokument.

7 Splošna priporočila za delo

Ni programa, ki bi deloval v prvem poizkusu. Kaj šele, da bi deloval pravilno. Zato je potrebno problem razdeliti na manjše dele (= module) in vsak modul posebej kodirati (pisati program), prevajati in sproti testirati. Posebej je to pomembno pri večji programih, kjer se kmalu izgubi preglednost nad celim programom. Pri izdelavi programa posvetite pozornost, da velja zaporedje:

- najprej mora program delovati
- nato mora delovati pravilno
- in na koncu (če je čas), mora delovati hitro

V navodilih za delo je veliko WWW naslovov, kjer lahko poiščete informacije ali programe. Če nimate dostopa do interneta, lahko strani pogledate na našem računalniku v avli fakultete ali na enem od računalnikov v laboratoriju LECAD (N-17). Na računalnikih v LECADu si lahko na disketo posnamete potrebne programe.

7.1 Kako testirati

Testiranje programa je pomemben del pisanja programa, ki je največkrat (zaradi pomanjkanja časa, tudi pri komercialnih programih) zanemarjen.
Testiranje je najbolje izvajati na posameznih delih programa. Izpis vrednosti spremenljivk med izvajanjem programa (v modulu, kjer je nekaj narobe) omogoča zelo hitro lokacijo napake. Pri sestavljanju programa (iz posameznih modulov) pa je najučinkoviteje, da se izpisujejo (oz. testirajo) vhodni in izhodni podatki modulov.

Za preverjanje pravilnosti VRML datotek je na strežniku LECAD laboratorija na voljo program **Live3D** (*ftp://ftp.lecad.uni-lj.si/pub/vaje/install/3dns32h.exe*), ki je dodatek za brskalnik (plug-in) Netscape Navigator.

7.2 Informacije in oddaja poročil

Govorilne ure so vsak dan, razen petka, od 12.00 do 13.00. Lahko komuniciramo tudi preko elektronske pošte. Naslovi so objavljeni na naši internet strani.

Pri oddaji programa morate poleg programa oddati tudi poročilo. Poročilo oddate na disketi, skupaj s programom, izpišete le prvo stran poročila. Poročilo in program pregledamo in nekaj dni po oddaji je zagovor programa. Zagovor poteka tako, da vsak študent dobi nalogo (= napisati kratek program - grafika, računanje ali kaj podobnega), ki jo mora narediti v določenem (predpisanem) času .

Za študente, ki v prvem roku niso opravili vaj, je na voljo še drugi rok, ki je navadno 14 dni za prvim rokom. Postopek zagovora je enak kot na prvem roku.

8 Literatura

- Michael E. Mortenson: Geometric Modeling, strani 366-369
- razni HTML dokumenti na straneh LECAD laboratorija

Napisano v LaTeX.