Seminarske naloge pri RPK

Leon Kos, LECAD

24. marec 1998

Povzetek

Predstavljene so naloge iz računalniške grafike in aplikativnih znanj s katerimi se opravljajo vaje programskega dela študija računalniško podprtega konstruiranja.

K izdelavi programa študentje pristopijo po vajah iz prostorskega modeliranja na delovnih postajah. Naloge so individualne in primerne le za enega slušatelja.

Kazalo

1 Uvod

Način opravljanja programskega dela vaj temelji na samostojnem delu študenta v sodelovanju s tutorjem posamezne naloge. Rezultat naloge je delujoč in potrjen program, poročilo v obliki HTML in predstavitev naloge pred avditorijem.

Posvetovanja o nalogi se opravljajo po potrebi v času govorilnih ur. Predstavljene naloge so razdeljene na aplikativne in na probleme iz računalniške grafike. Aplikativne naloge uporabljajo že pridobljena strojniška znanja, ki jih uporabimo za izdelavo programa. Lahko pa so naloge ozko usmerjene na določene probleme, ki jih rešujejo inženirji pri vsakdanjem delu. Take naloge so tudi s področja podatkovnih baz ali splošnega modeliranja objektov. Večina nalog je s področja računalniške grafike pri katerih se je potrebno s problemom šele seznaniti in ga ustrezno rešiti ni preliti v programsko kodo. Za reševanje nalog s tega področja dobijo študentje osnove na predavanjih RPK.

Za izvedbo naloge se uporablja programski jezik po izbiri in primernosti.Nekatere naloge vsebujejo priporočila o izbiri jezika, v katerem bi bilo nalogo izdelati najenostaveje in dovolj učinkovito. Priporoča se znanje jezika C. Možna pa je uporaba tudi jezikov kot so Javascript, Java, Fortran, Pascal, Perl, ...

Cilj naloge je naveden, poleg tega pa je ob zaključku potrebno dostaviti še poročilo, ki mora biti v elektronski obliki z ustrezno opremljeno. (Besedilo, enačbe, slike). Ciljna datoteka poročila mora biti v obliki HTML, ki pa se jo lahko dobi tudi s konverzijo ustreznega urejevalnika besedil (Word, Wordperfect, LaTeX, ...). Ocena se dobi ob izpolnitvi cilja naloge in oceni kvaliteta izdelave.

Poročilo naj ima standardno zgradbo s povzetkom (*Abstract*) v angleškem jeziku. Potrebna navodila za izdelavo poročil se nahajajo tudi na WWW strežniku laboratorija LECAD[1]

Predstavitev izdelanih nalog, ki so bile popreje pregledane in potrjene pri tutorju bodo:

Prvi rok 29. junija

Drugi rok 7. septembra

2 Programiranje

Predstavljeni so osnovni prijemi, s katerimi lahko izdelamo naloge.

2.1 Površine v formatu DXF

Data eXchange File je specifikacija podana pri firmi Autodesk, vendar jo je povzelo večina proizvajalcev. Datoteka, ki je tako generirana se lahko uvozi v različne 3D programske pakete *AutoCAD, Ideas, Microstation, Alias, ...*

Primer 3D DXF datoteke ki vsebuje eno samo površino (3DFACE):

```
  0
SECTION
  2
ENTITIES
  0
3DFACE
  8
0
 10
-0.500000
 20
0.000000
 30
0.000000
 11
-0.500000
 21
0.000000
 31
0.000000
 12
-0.498459
 22
-0.006137
 32
0.038747
 13
-0.498459
 23
0.000000
 33
0.039230
  0
ENDSEC
  0
EOF
```

3DFACE je sestavljen iz štirih točk in korordinat *x, y, z.* DXF označbe `10, 20 in 30` podajajo točko 0; `11, 21 in 31` točko 1 in tako naprej.

[1] http://www.lecad.uni-lj.si/documents/

2.2 Javascript

Javascript je programski jezik, ki se interpretira v brskaliku kot je Nestscape Navigator ali Microsoft Internet Explorer. Ukazne strukture so enake jeziku C. Program se piše v HTML dokumentu pod sekcijo script. Z njim je možno kreirati aktivne HTML dokumente, ki se sestojijo iz slikic, besedila, gumbov, vnosnih polj...

Primer Javascript programa, ki se ga lahko izvede v brskalniku in ob pritisku na gumb "Dolžina" odpre novo okno in izračuna dolžino vektorja, ki ga podamo v treh poljih s koordinatami x, y in z:

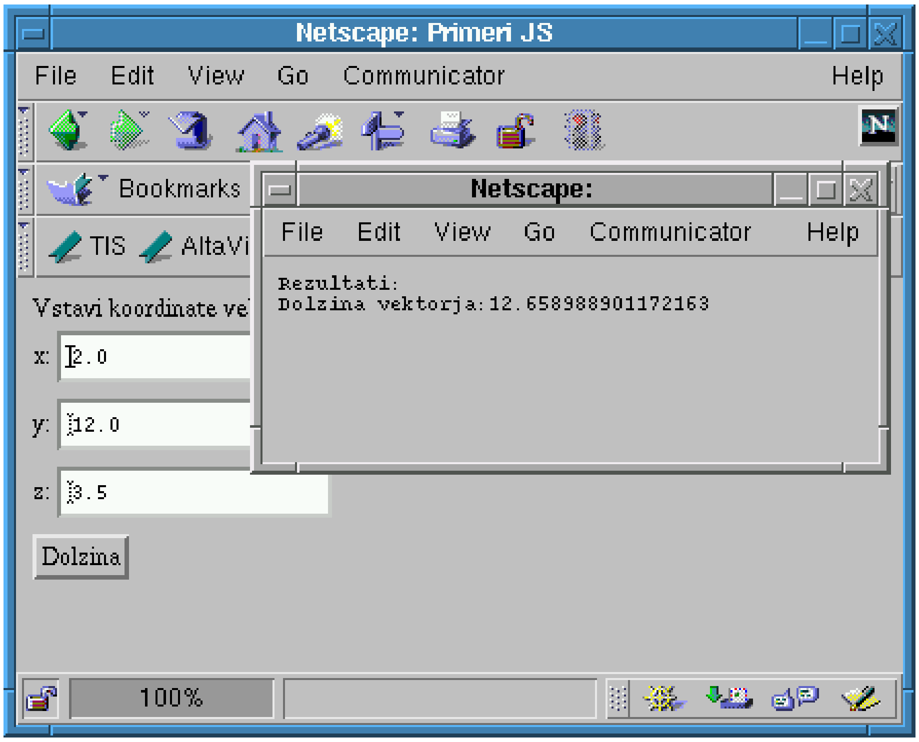
```
<HTML>
<HEAD>
<TITLE>Primeri JS</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function sqr(x)  // kvadrat
{
   return x*x;
}

function dolzina(vektor)
{
  var l = 0.0;
  okno = open("", "Rezultati", // novo okno
        "width=300,height=100,menubar=yes,scrollbars=yes");
  okno.creator = self;

  for (var i= 0 ; i< vektor.length ; i++)
    {
         l += sqr(parseFloat(vektor[i].value))
    }

   l = Math.sqrt(l);

  okno.document.open("text/plain");
  rez=okno.document
  rez.write("Rezultati:\n");
  rez.write("Dolzina vektorja:" + l + "\n");
  okno.document.close();
}
</SCRIPT>
</HEAD><BODY>
<FORM>
Vstavi koordinate vektorja:<BR>
x:<INPUT TYPE="text" NAME="v1" VALUE="0"><BR>
y:<INPUT TYPE="text" NAME="v1" VALUE="0"><BR>
z:<INPUT TYPE="text" NAME="v1" VALUE="0"><BR>
<INPUT TYPE="button" VALUE="Dolzina"
       ONCLICK="dolzina(this.form.v1)" >
</FORM></BODY></HTML>
```



Slika 1: Primer Javascript programa ki izračuna dolžino vektorja in odpre novo okno z rezultati

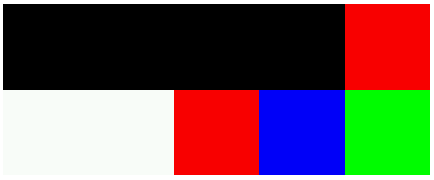
Zelo priročen uvod v Javasvcript je Voodoo's Introduction to Javascript[2] avtorja Stefana Kocha, ki se dobi tudi v elektronski obliki. Obstajajo tudi navodila za Javascript 1.2[3] kot tudi najpogostejša vprašanja FAQ[4].

2.3 Rastrske slike

Generiranje rastrskih slik je lahko zelo zahtevno, saj obstajajo različni formati kot so BMP, GIF, TIFF, PNG, za katere je potrebno generirati bitno datoteko v točno določenem formatu. Obstaja pa tudi format PPM (Portable Pixmap Format) ki ga lahko zapišemo v obliki ASCII v katerem koli programskem jeziku, ki zna pisati datoteke.

2.4 Portable PixMap

Primer zelo majhne rastrske datoteke širine 5 točk in višine 2 točki je prikazan na sliki 2.



Slika 2: Rastrska slika primera v PPM zapisu

Datoteka `primer.ppm` ki je bila prikazana na sliki 2 je naslednja:

```
P3
5 2
255
0 0 0     0 0 0     0 0 0
0 0 0   255 0 0   255 255 255
255 255 255
255 0 0   0 0 255   0 255 0
```

Prva vrstica ima magični znak P3, ki je potreben za razpoznavo tipa datoteke. V drugi vrstici je označba 5 točk široke slike in 2 točki visoke slike. Skupaj je to 10 točk (pikslov). Število 255 označuje maksimalno velikost svetlosti ki je v razponu od 0 do 255. Za vsako točko se določi trojke števil za rdečo, zeleno in modro barvo (RGB). Zato sledi 3*10=30 števil. Črna barva ima vrednost (0 0 0), bela pa (255 255 255). Rastrska slika se začne sestavljati v zgornjem levem kotu. V okolju X-Windows lahko gledamo PPM datoteke s programom `xv`, v okolju MS-windows pa s programom `Paint Shop`.

[2] http://rummelplatz.uni-mannheim.de/ skoch/js/
[3] http://home.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html
[4] http://developer.netscape.com/support/faqs/champions/javascript.html

2.5 Windows BitMaP

V okolju Windows lahko zapišemoemo tudi datoteko BMP, ki jo razume večina programov (Panitbrush, Slikar, ...). Žal je format zahtevnejši in je zato primeren le za pisanje iz naših programov. Za branje pa bi bil primer bolj kompleksen, saj obstaja vrsto podformatov ter kompresij rastrskih podatkov. Pisanje datoteke se najlažje izvede v jeziku C, v Fortranu pa standardno to ni mogoče.

Primer programa, ki zapiše 24-bitno RGB datoteko:

```
/* Primer pisanja Windows BMP datoteke
   za LITTLE-ENDIAN (PC) racunalnike  LK98*/

#include <stdio.h>

#define WIDTH  28     /* mnogokratnik stevila 4 */
#define HEIGHT 25     /* visina slike */

#define RED 2
#define GREEN 1
#define BLUE 0

int
main()
{
  struct
    {
      unsigned char _[2];
      unsigned long FileSize : 32;
      unsigned char __[12];
      unsigned long ImageWidth : 32;
      unsigned long ImageHeight : 32 ;
      unsigned char ___[8];
      unsigned long SizeOfBitmap : 32;
      unsigned char ____[16];
    } static header = {
    {'B', 'M'}, 0, { 0,0,0,0, 0x36, 0,0,0, 0x28, 0,0,0},
    0, 0, { 1,0, 0x18,0, 0,0, 0,0}, 0,
    {0x6d, 0x0b, 0, 0, 0x6d, 0x0b, 0,0,0,0,0,0,0,0,0,0} };

  FILE *fbmp;
  int i, j;
  unsigned char img[HEIGHT][WIDTH][3];

  /* Nastavimo glavo BMP datoteke */
  header.ImageWidth = WIDTH;
  header.ImageHeight = HEIGHT;
  header.SizeOfBitmap = WIDTH * HEIGHT * 3;
  header.FileSize =  header.SizeOfBitmap + 0x36;

  /* Celotna slika je crna */
  for(i = 0; i < HEIGHT; i++)
```

```
    for(j = 0; j < WIDTH; j++)
        img[i][j][0] = img[i][j][1] = img[i][j][2] = 0;
  /* razen ene rumene pike spodaj levo */
  img[0][0][RED] = 255 ; img[0][0][GREEN] = 255;

  fbmp = fopen("primer.bmp", "wb");
  fwrite(&header, 0x36,  1, fbmp);
  fwrite(img, 3, WIDTH*HEIGHT, fbmp);
  fclose(fbmp);

  return 0;
}
```

2.6 PHIGS

V laboratoriju LECAD je razvita knjižnica podprogramo, ki emulira osnovne ukaze grafičnega jezika PHIGS za risanje črt v okolju windows in z uporabo prevajalnika DJGPP, ki ga lahko študentje dobijo brezplačno skupaj z knjižnic phigs in ustreznimi navodili za njeno uporabo. Prevajalnik DJGPP omogoča prevajanje v jeziku Fortran ali C.

2.7 Grafika v Windows

Programiranje grafike za okolje Windows je zahtevno, vendar je za osnovno risanje slikic dovolj že enostaven program, ki ga lahko prevedemo s prevajalnikom za Windows. Obstaja tudi brezplačen prevajalnik Cygnus[5], ki zna prevajati win32 kodo. Prevajalnik je možno dobiti tudi na našem strežniku `ftp.lecad.uni-lj.si` na področju `/pub/vaje/install`.

Program prevedemo z ukazom: `gcc primer.c -o primer.exe -lgdi32 -luser32`

```
#include <windows.h>

LRESULT FAR PASCAL AppWndProc(HWND hWnd, UINT msg, WPARAM wParam,
                              LPARAM lParam);

HWND hAppWnd;

int WINAPI
 WinMain(HANDLE hInstance, HANDLE hPrevInstance, LPSTR
     lpCmdLine, int nCmdShow)
{
    MSG msg;
    static const char
     szWndClassName[] = {"MYWNDCLASS"};

    if (hPrevInstance == NULL) {
      WNDCLASS wc;
    wc.lpszClassName = szWndClassName;
    wc.hInstance = hInstance;
```

[5]ftp://sunsite.fri.uni-lj.si/pub/gnu-win32/latest

```
    wc.style = CS_HREDRAW | CS_VREDRAW;
    wc.lpfnWndProc = AppWndProc;
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    wc.hIcon = NULL;
    wc.hCursor = NULL;
    wc.hbrBackground = (HBRUSH) GetStockObject(WHITE_BRUSH);
    wc.lpszMenuName = NULL;
    if (!RegisterClass(&wc))
        return FALSE;
    }
    // Create our Main Application Window
    hAppWnd = CreateWindow(szWndClassName, "Test Application",
              WS_OVERLAPPEDWINDOW,  // Style of Window
              CW_USEDEFAULT, CW_USEDEFAULT, 400, 300,
              NULL, NULL, hInstance,
              NULL);

    ShowWindow(hAppWnd, nCmdShow);
    UpdateWindow(hAppWnd);

    while (GetMessage(&msg, NULL, 0, 0)) {
      TranslateMessage(&msg);   /* translates virtual key codes    */
      DispatchMessage(&msg);    /* dispatches message to window    */
    }

    return (int) msg.wParam;    /* return value of PostQuitMessage */
}


LRESULT FAR PASCAL
 AppWndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    // this is where we receive all message concerning this window
    // we can either process a message or pass it on to the default
    // message handler of windows

    switch (msg) {
    case WM_CREATE:
    // Initialize Window
    // this message is received before the window is displayed
    break;
    case WM_PAINT:
    {
        static LOGPEN lpBlack =
        {PS_SOLID, 1, 1, RGB(0, 0, 0)}, lpWhite =
        {PS_SOLID, 1, 1, RGB(255, 255, 255)};

        PAINTSTRUCT ps;
        HDC hdc;
```

```
        RECT rect;
        HPEN hPenBlack, hPenWhite;
        int i;

        // Get the handle to the Windows's Display Context
        hdc = BeginPaint(hWnd, &ps);
        // Now Paint the Windows's contents
        // PaintAppWindow(hWnd,hdc,iPaintCount);
        // Call EndPaint to release the DC and validate

        hPenBlack = CreatePenIndirect(&lpBlack);
        hPenWhite = CreatePenIndirect(&lpWhite);

        // SetMapMode (hdc, MM_ANISOTROPIC);
        GetClientRect(hWnd, &rect);
        // SetViewportExtEx (hdc, rect.right, rect.bottom, NULL);
        // SetWindowExtEx(hdc, 10, 4, NULL);

        // Tu ri\v{s}emo \v{c}rto
        SelectObject(hdc, hPenBlack);
        MoveToEx(hdc, 1, 1, NULL);
        LineTo(hdc, 9, 1);

        DrawText(hdc, "Hello, Windows", -1, &rect,
             DT_SINGLELINE | DT_CENTER | DT_VCENTER);

        EndPaint(hWnd, &ps);
    }
    break;
    case WM_DESTROY:
    PostQuitMessage(0); // Terminate Application

    break;
    default:
    // We didn't process the message so let Windows do it
    return DefWindowProc(hWnd, msg, wParam, lParam);
    }
    // We processed the message and there
    // is no processing by Windows necessary
    return 0L;
}
```

2.8 VRML

VRML jezik za opis prostorskih objektov je ASCII datoteka, ki jo lahko gledamo v brskalniku ali samostojnem predvajalniku. Primeren program na okolju Windows je Cosmo Player[6], ki se dobi brezplačno na WWW strežniku Silicon Graphics[7]. Izdelati je potrebno primer datoteke `lecad.wrld` z 3D modelom

[6] http://www.sgi.com/Fun/Free_webtools.html
[7] http://www.sgi.com/

črk napisa LECAD in uporabo osnovnih primitivov. Poročilo naj vsebuje tudi kratek opis jezika VRML. Navodila in opis jezika obstaja tudi na strežniku laboratorija LECAD[8]

```
#VRML V2.0 utf8
Transform {
  children [

    DirectionalLight {        # First child
        direction 0 0 -1      # Light illuminating the scene
    }

    Transform {               # Second child - a red sphere
      translation 3 0 1
      children [
        Shape {
          geometry Sphere { radius 2.3 }
          appearance Appearance
             material Material { diffuseColor 1 0 0 }   # Red
        }
      ]
    }

    Transform {               # Third child - a blue box
      translation -2.4 .2 1
      rotation     0 1 1  .9
      children [
        Shape {
          geometry Box {}
          appearance Appearance
             material Material { diffuseColor 0 0 1 }  # Blue
        }
      ]
    }

  ] # end of children for world
}
```

3 Aplikativne naloge

3.1 Parametrični program za generiranje sestavnih delov okenskih profilov

Parametrični program vsebuje generiranje geometrije za izbrane osnovne plastične profile. Program za znane oblike (štirikotnik, trapez, trikotnik, polkrog) generira datoteko z opisom površin izbranega profila v obliko DXF. Pri tem je potrebno pravilno orientirati normale 3DFACE elementov.

[8] http://www.lecad.uni-lj.si/documents/other/vrml2.0/Overview.html

Program naj omogoča grafični izbor željenega profila iz katerega se bo izdelal okvir, obliko okvira (pravokotnik, trapez, trikotnik, palica, ...) in dimenzije. Po vnosu potrebnih podatkov naj se izpiše DXF datoteka, ki se jo nato lahko prenese iz brskalnika v odlagališče (*clipboard*).

Priporočen jezik za izdelavo je JavaScript.

3.2 Parametrični program za izdelavo vrat s podboji

Podobno kot pri 3.1 je tu potrebno izdelati program, ki bo izbiral levo ali desno odpiranje, vprašal oz predlagal standardne dimenzije in generiral 3D geometrijo skupaj z 2D tlorisnim pogledom za klasična notranja vrata. Oblika podboja naj bo suhomontažna ali mokromontažna.

Priporočen jezik za izdelavo je JavaScript.

3.3 Nosilec z I profili

Za podano poljubno obremenitev nosilca v prostoru je potrebno izračunati obremenitve in izrisati momente v posameznih ravnivah , kot tudi skupne. Program bere vhodno datoteko s podatki in preračuna momente glede na položaj sil in momentov ter obremenitve. Rezultate izračuna izpiše in izriše v 3D. Za izdelavo lahko uporabimo knjižnico PHIGS.

3.4 Geometrijske karakteristike prereza

Interaktivni program, s katerim izračunamo geometrijske karakterisike kot so površina, težišče, vztrajnostni momenti z glavnimi osmi: Program v Javascriptu v katerem tabelarično izračunamo karakteristiko prereza generira tudi 3D VRML datoteko za kontrolo vhodnih podatkov kot tudi rezultatov.

3.5 Hexapod

Izdelati je potrebno program, ki bo prikazal direktno kinematiko stroja za frezanje v obliki mehanizma Hexapod[9]. Program naj bo zmožen animacije in spreminjanja dolžin segmentov s pritiskom na tipke (npr. F1 do F12). Najlažje je to izdelati s programom v C-ju in Windows ali OpenGL knjižnici.

3.6 Tlačne vzmeti

Potrebno je izdelati izračun tlačnih vzmeti po ustrezni literaturi (npr. DIN 2089-1) in izdelati DXF datoteko s produkcijsko sliko vzmeti za uvoz v 2D modelirnik. Priporočen jezik za izdelavo je Javascript.

3.7 Torzijske vzmeti

Podobno kot v 3.6, le da so tu torzijske vzmeti in DIN 2091 ali podoben postopek.

[9] http://www.hexel.com/

Slika 3: Hexapod frezalni stroj

3.8 Statisticna obdelava kontrole vstopa

Za statistiko prehoda vrat se lahko generira datoteka v obliki HTML, ki se dnevno osvežuje. Računalnik beleži vse dogodke v ASCII datoteki, ki jo s poljubnim programom interpretiramo in napišemo datoteko v formatu HTML, tako da jo lahko strežnik prikaže. Program je možno napisati v poljubnem jeziku, saj gre le za branje in pisanje datoteke. Statistične grafe za dnevni, tedenski in mesečni prehod in variacije je potrebno izdelati v jeziku HTML in sestavljanjem slikic (bar) v binarni obliki. Primer za takšno generiranje je program Analog za statistiko WWW strani. Pomoč in informacije: Janez Krek.

Primer vhodne datoteke je naslednji:

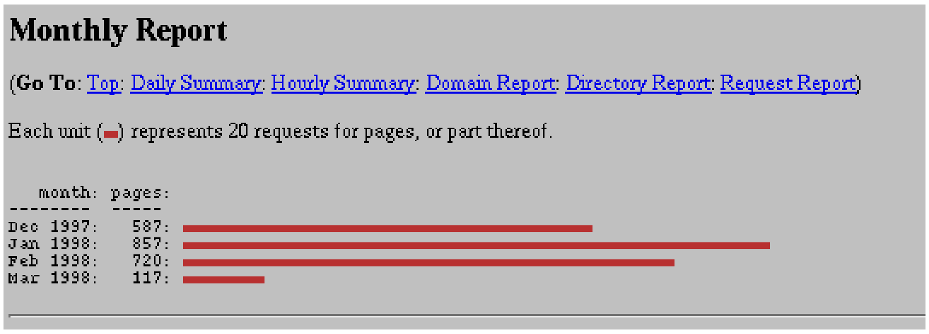
```
....
Mar 12 15:54:40 : Computer: door open from leon@achilles.
Mar 12 15:57:18 : Keyboard: used INSIDE keyboard, typed code 4.
Mar 12 15:59:06 : Keyboard: used OUTSIDE keyboard, typed code 2.
Mar 12 16:01:22 : Keyboard: used INSIDE keyboard, typed code 4.
Mar 12 16:53:48 : Key 00000034C843 (dr. Joze DUHOVNIK - 8): Entrance.
Mar 12 16:55:59 : Key 00000034C843 (dr. Joze DUHOVNIK - 8): Leaving.
Mar 12 17:34:00 : Key 00000030D6E8 (Leon Kos - 8): Leaving.
Mar 12 18:18:46 : Key 000000194C98 (Snazilka - 8): Entrance.
Mar 12 18:22:05 : Key 000000194C98 (Snazilka - 8): Leaving.
Mar 12 18:29:23 : Key 0000004349B2 (Janez KREK - 8): Leaving.
Mar 12 18:35:28 : Key 000000194C98 (Snazilka - 8): Leaving (not in the lab)
Mar 12 18:36:00 : Key 000000194C98 (Snazilka - 8): Leaving (not in the lab)
Mar 12 18:36:14 : Key 000000194C98 (Snazilka - 8): Entrance.
Mar 12 18:36:24 : Key 000000194C98 (Snazilka - 8): Entrance.
Mar 12 19:26:47 : Keyboard: used OUTSIDE keyboard, typed code 2.
Mar 12 20:58:10 : Keyboard: used INSIDE
keyboard, typed code 4.
```

...

Izhodna datoteka naj bi uporabljala sestavljanje osnovnih palčk in s tem graf. Primer izhodne datoteke, ki je prikazana na sliki 4 je naslednji:

```
<HTML>
<h2><a NAME="Monthly">Monthly Report</a></h2>

<p>Each unit (<tt><img src="/images/bar1.gif" alt="+"></tt>)
 represents 20 requests for pages, or part thereof.
<pre width=65><tt>
   month: pages:
--------  -----
Dec 1997:   587: <img src="/images/bar16.gif"><img src="/images/bar8.gif">
      <img src="/images/bar4.gif"><img src="/images/bar2.gif">
Jan 1998:   857: <img src="/images/bar32.gif"><img src="/images/bar8.gif">
     <img src="/images/bar2.gif"><img src="/images/bar1.gif">
Feb 1998:   720: <img src="/images/bar32.gif"><img src="/images/bar4.gif">
Mar 1998:   117: <img src="/images/bar4.gif"><img src="/images/bar2.gif">
</tt></pre><hr>
</HTML>
```



Slika 4: Del HTML dokumenta z izrisano statistiko

3.9 Gredi

Statični in trdnostni izračun gredi z naborom obremenitev. Za izbrane klasične podpore in položaje sil in momentov je potrebno izrisati in izračunati gred. Priporočen je Javascript in VRML. Če pa se bo uporabil PHIGS, mora program omogočati ustrezne 3D transformacije in izris.

3.10 Ležaji

Izračun ležajev v Javascriptu. Za podani sili F_r, F_a, srednjo hotrost, faktor obratovalnih pogojev a_{23}, delovno temperaturo, viskoznost in % delovne dobe je potrebno izračunati delovno dobo. Program naj omogoča tudi možnost izračuna dinamične nosilnosti z izračunom ekvivalentne obremenitve. Program naj bo v jeziku Javascript.

3.11 Jermenica

Izris žičnega modela z upoštevanjem skupine prestav in konstantno medosno razdaljo. Za podana razmerja prestav je potrebno določiti premere gonilnega in gnanega koluta, tako da je medosna razdalja za vse prestave enaka. Klinasti jermen. Izris s Phigsom. Jezik F77 ali drugi.

3.12 Podatkovna baza

Baze podatokov so osnova za hranjenje v urejeni obliki. Format `DBF` ali DBase File ježe zelo star format zapisa datoteke. Izdelati je potrebno knjižnico podprogramov, v s katero bo mogoče zapisati in prebrati podatke v DBF formatu. Knižnica podprogramov naj bo zmožna iskanja v izbranem polju. Indeksiranje ni potrebno. Ker je format datoteke, ki je opisan v [4] binarni je možna izdelava le v jeziku C. Predvideno število podprogramov je okoli 10 in nekaj strani izvorne kode. Za primer aplikacije knjižnice, naj bi se uporabila baza študentov.

3.13 Java razredi za komunikacijo s strežnikom Hylafax

Za strežnik cad.lecadnet s TCP portom št: 4559 obstaja enostaven protokol kot tudi izvorna koda s primerom komunikacije. Za neodvisnost od platforme se lahko uporabi jezik Java, ki ima vgrajeno mrežno komunikacijo. Tako je možno odpreti mrežno datoteko in poslati podatke strežniku, ki sprejete podatke interpretira in pošlje datoteko naprej do strežnika faksov. Primer programa, ki uporablja paketni protokol UDP za odpiranje vrat v laboratoriju LECAD je naslednji:

```
// Import the package names used.
import java.io.*;
import java.net.*;
import java.util.*;
import java.text.*;

/**
 * This is an application to obtain the GATE
 * opening  from
 * remote system via UDP and.
 * @author Leon  Kos
 * @date jan 1998
 * @version 0.1
 */
public class gate {

private static final int GATE_PORT = 8909;

 // A socket to send and receive datagrams.
 DatagramSocket GATESocket = null;
 // An array of addresses to the machines
 private InetAddress remoteMachine;

 /**
  * This method starts the application.
  * @param args Command line arguments
  */
 public static void main(String[] args) {
    if (args.length < 1) {
```

```
      System.out.println(
         "Usage: gate host");
      System.exit(1);
   }

   gate door = new gate(args[0]);

   System.out.println("Vrata odprta.");
   System.exit(0);            // Exit.
}

/**
 * The constructor looks up the remote
 * hosts and creates a UDP socket.
 * @param hosts The hosts to contact.
 */
public gate(String host) {
 try {
   remoteMachine = InetAddress.getByName(host);
  } catch(UnknownHostException excpt) {
     remoteMachine = null;
      System.err.println("Unknown host " +
                host + ": " + excpt);
  }

try {
    GATESocket = new DatagramSocket();
} catch(SocketException excpt) {
  System.err.println(
    "Unable to bind UDP socket: "
      +  excpt);
    System.exit(1);
 }
   // Perform the UDP communications.
   openGate();
}

public void openGate() {
  DatagramPacket GATEQuery;
  DatagramPacket response;
  byte[] buffer;
  int datagramsSent = 0;

   // Send out a small UDP datagram,
   // asking it to respond with its GATE.

  if (remoteMachine != null) {
    try {
      buffer = new byte[30];
      String from = unknown@" +
      InetAddress.getLocalHost().getHostName();
      buffer = from.getBytes();

      GATEQuery = new DatagramPacket(
```

```
            buffer, buffer.length,
            remoteMachine, GATE_PORT);
        GATESocket.send(GATEQuery);
        datagramsSent++;
    } catch(IOException excpt) {
      System.err.println(
        "Unable to send to " +
        remoteMachine + ": " + excpt);
    }
   }

   // Close the socket.
   GATESocket.close();
   GATESocket = null;
}
```

Pomoč pri izdelavi in programiranju nudi Janez Vrhovec.

3.14 Dimnik

Potrebno je poiskati ustrezne specifikacije (npr. DIN 4705) za izračun potrebnega preseka za podane vhodne podatke. Program naj oceni tudi čas potreben za segretje ali ohlajanje tuljave glede na vhodno temperaturo dimnih plinov in zunanjo temperaturo. Program naj generira geometrijo z DXF datoteko za uvoz v modelirnik.

3.15 Sestavi omar

Za podane osnovne oblike omar je potrebno generirati DXF datoteko sestava na podanih mestih in z določeno orientacijo. Priporočen jezik za izdelavo je Javascript.

4 Računalniška grafika

4.1 VRML – Virtual Reality Modeling Language

V jeziku VRML je možno opisat 3D predmete, ki ji lahko gledamo v brskalniku s uporabo ustreznega programa za gledanje. Primeren program na okolju Windows je Cosmo Player[10], ki se dobi brezplačno na WWW strežniku Silicon Graphics[11]. Izdelati je potrebno primer datoteke `lecad.wrld` z 3D modelom črk napisa LECAD in uporabo osnovnih primitivov. Poročilo naj vsebuje tudi kratek opis jezika VRML. Navodila in opis jezika obstaja tudi na strežniku laboratorija LECAD[12]

4.2 Knjižnica za risanje v jeziku postscript

Izdelava podprogramov v jeziku C ali Fortran za risanje v postscriptu, ki je grafični jezik za večino sodobnih tiskalnikov. Programi morano uporabljati geometrijske transformacije za skaliranje, transliranje in rotiranje. Primer za podprogram, ki riše črto od točke (10,10) do točke (20,400) je naslednji:

```
10 10  moveto
20 400 lineto
showpage
```

[10] http://www.sgi.com/Fun/Free_webtools.html
[11] http://www.sgi.com/
[12] http://www.lecad.uni-lj.si/documents/other/vrml2.0/Overview.html

Za preverjanje grafičnega izhoda upograbimo interpreter jezika postscript *ghostscript*, ki ga lahko dobimo na množici strežnikov. Za razumevanje jezika postscript je dovolj *First Guide to Postscript*, ki ga dobite v laboratoriju pri Tomažu Kolšku.

4.3 Osenčene krogle

Risanje osenčenih krogel z svetlobnim modelom PHONG[10] pri katerem se svetlost točke določa po formuli

$$I = I_a K_a r + I_i[k_d(\vec{L} \cdot \vec{N}) + k_s(\vec{N} \cdot \vec{H})^n] \tag{1}$$

kjer je I_a intenziteta abmienta, $\vec{L}$ vektor luči, $\vec{N}$ normala ploskve, $\vec{H}$ vektor očesa.

Narisati je potrebno serijo krogel, z variranjem koeficienta odbleska k_s in širino odbleska n.

Za izdelavo naloge je možno je uporabiti tudi jezik fortran, le da je potrebno pisati rezultat kot rastrsko sliko v ASCI formatu PPM (portable Pixmap Format). Opis formata PPM je v enciklopediji grafičnih formatov, ki jo lahko dobite v laboratoriju.

4.4 Odbojni model Cook-Torrance

Podobno kot zgoraj, le da je model bolj realističen in enačba drugačna[10]. Za odsevno površino izberemo ravnino za katero izračunamo presek med vektorjem očesa in ravnino. Zahteve za program so podobne koot pri Phongu.

4.5 Skrivanje črt

Napisati je potrebno program, ki bere trikotnike in jih z metodo spanning scan uredi ter izračuna skrite linije, ter prikaže na na zaslonu. Za grafiko se lahko uporabi knjižbica za PHIGS v jeziku C ali Fortran.

4.6 Bézierjeve krivulje

Risanje parametričnih krivulj je uporabljeno v večini sodobnih programov. Najenostavnejša je Bézierjeva krivulja, ki jo v prostoru popišemo z enačbo

$$\vec{P} = [\,u^3 \quad u^2 \quad u \quad 1] \begin{bmatrix} -1 & -3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \vec{P}_0 \\ \vec{P}_1 \\ \vec{P}_2 \\ \vec{P}_3 \end{Bmatrix} \tag{2}$$

kjer so $\vec{P}$ vektorji kontrolnih točk, u pa parameter v mejah od 0 do 1. Izdelati je potrebno program, ki bo izrisoval krivulje in imel možnost urejanja. Za grafiko se lahko uporabi knjižnica za PHIGS v jeziku C ali Fortran.

4.7 Sence

Program za risanje senc točkovnega izvora svetlobe in upoštevanjem umbre in pentumbre. Za like v prostoru je potrebno določiti projekcijo na ravnino. Točkovni izvor projiciramo skozi trikotnik ali štirikotnik poljubno orientiran v protoru na ravnino. Izdelati je potrebno program, ki vsebuje ustrezne podprograme za projekcijo izvora v obliki štirikotnika na ravnino. Prikaz s knjižnico PHIGS.

4.8 Voxel editor

Prostorske točke lahko pokažemo ravno tako kot točke na zaslonu. Za vrtenje v prostoru se lahko uporabi algoritem Marching-Cube[10]. Izdelati je potrebno osnovo za prikaz $256 \times 256 \times 256$ voxlov veliko kocko in predstaviti ustrezne algoritme za prikaz prostornine.

4.9 Rezanje (clipping) v homogenih koordinatah

Za rezanje odvečnih elementov kot so črte v prostoru se upororablja algoritemfoley90:_comput_graph, ki je posebno primeren za homogeni prostor. Izdelati je potrebno podprogram, ki pri paralelni projekciji določi odrez črt in vse skupaj za enostaven primer pokaže na zaslonu. Za prikaz se lahko uporabi PHIGS.

4.10 Globinski pomnilnik - Z buffer

Potrebno je izdelati 3D primer za globinski pomnilnik Z buffer s knjižnico PHIGS. Primer za 3D objekt, ki ga globinsko uredimo je tudi krogla popisana s kvadratnimi ploskvicami.

4.11 Backface cullling

Izločanje površin[3, 10] glede na orientacijo. Za primer koluta je potrebno generirati površinice, jim določiti normale in opraviti izločanje ter prikaz.

4.12 Kvadriki in superkvadriki

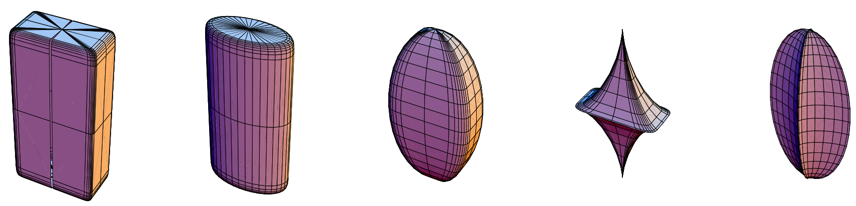
Superkvadriki [1] so razširitev osnovnih enačb kvadrikov definiranih kot sferični produkt v parametričnem 3D prostoru. Implicitna enačba Piet-Heinove superelipse je:

$$|x|^{\frac{2}{\varepsilon}} + |y|^{\frac{2}{\varepsilon}} = 1 \qquad \varepsilon = const \tag{3}$$

ki jo lahko zapišemu tudi v parametrični obliki kot

$$x = \sin^{\varepsilon}\theta \qquad y = \cos^{\varepsilon}\theta \qquad -\pi \leq \theta < \pi \tag{4}$$

kjer je ε vzhodno/zahodni ali parameter zaokrožitve robu.



Slika 5: Primeri elipsoidov superkvadrikov

Razšitritev superelips v prostor razdeli enačbe na elipsoide (slika 5) in toroide. Za elipsoide je parametrična enačba definirano kot:

$$\begin{aligned} x(u,v) &= a_x c(v,n) c(u,e) \\ y(u,v) &= a_y c(v,n) s(u,e) \\ z(u,v) &= a_z s(v,n) \\ -\pi/2 \leq v \leq \pi/2 \qquad &, \qquad -\pi \leq n \leq \pi \end{aligned} \tag{5}$$

in podobno za toroide

$$
\begin{aligned}
x(u,v) &= a_x c_T(v,n,\alpha) c_T(u,e) \\
y(u,v) &= a_y c_T(v,n,\alpha) s(u,e) \\
z(u,v) &= a_z s(v,n) \\
-\pi \leq v \leq \pi \quad &, \quad -\pi \leq n \leq \pi
\end{aligned} \tag{6}
$$

Podfunkcije uporabljene v zgornjih enačbah eq:sqpar in 6) so definirane kot:

$$
\begin{aligned}
c(v,n) &= sgn(\cos v)|\cos v|^n \\
c_T(v,n,\alpha) &= \alpha + c(v,n)\ , \qquad \alpha > 1 \\
s(v,n) &= sgn(\sin(v))|\sin(v)|^n
\end{aligned} \tag{7}
$$

Izdelati je potrebno program, ki bo prikazal žični model kvadrikov in superkvadrikov z možnostjo izbire osnovnih parametrov. Izris v PHIGS-u.

4.13 Presek žarkov s kroglami

Izpeljati je potrebno presek črte (žarka) s kroglo in izdelati rekurziven program, ki bo narisal v prostoru odboj žarka z večjim številom krogel. Žični model in izris v PHIGS-u.

4.14 RGB barvni model

Izris bravnega modela kocke (Red - Green - Blue) v različnih orientacijah v rastrsko datoteko z uporabo homogenih transformacij.

4.15 HSV barvni model

Izris barvnega modela Hue - Saturation - Value s presekom ravnine in izrisom v rastrsko datoteko. Imamo dva možna preseka: stranski ris in tloris. Uporabnik naj izbere pogled in parametre.

4.16 Raztros črnobelih odtenkov

Z metodo difuzije napake[9] (Floyd-Steinberg) lahko na črnobelih tiskalnikih in zaslonih ustvarimo učinek sivin, tako da variramo gostoto pojavljanja črnih pik. Izdelati je potrebno program, ki prebere rastrsko datoteko s sivimi odtenki in z metodo difuzije napake ustvari črnobelo sliko enake ločljivosti.

4.17 Urejen raztros

Podobno kot v 4.16, le da tu uporabimo urejen raztros [9] z matriko v štirikotni ali šestkotni obliki.

4.18 Mrezenje v 2D - Vornoi

Za množico točk izdelajte program, ki bo naredil optimalno trikotnižko mrežo v 2D z uporabo Vornoi diagramov[5] in njegovega duala - Delaunay triangulacija. Z uporabo knjižnice PHIGS je potrebno izrisati generirano mrežo trikotnikov. Obstajajo tudi primeri v programskem jeziku java.[13]

[13] http://ra.cfm.ohio-state.edu/grad/zhao/algorithms/algorithms.html

4.19 Proceduralne teksture

Teksture so slike, ki jih lepimo na površine in s tem dobimo vtis materiala za objekt, ki ga površina opisuje[6]. Možna sta dva pristopa: skaniranje ali procedura. S proceduro popišemo materiale tako, da podamo ustrezne parametre, nato pa s algoritmom generiramo barvo posamezne točke v prostoru. Izdelati je potrebno program, ki bo generiral teksturi lesa in marmorja.

4.20 Phigs viewing matrix

Po postopku, kot je opisan v [3] je potrebno izdelati knjižnico podprogramov za projekcijo 3D objektov na ravnino.

4.21 Razpoznavanje daljic

Za podano rastrsko datoteko na kateri je narisano več daljic, je potrebno razpoznati dolžino in orientacijo daljice in jih zapisati v vektorski obliki (začetek - konec). Izbira algoritma je poljubna.

Primer je (x/y) v (a/b) transformacija: Imejmo točke v ravnini podane s koordinatama x in y. Če izberemo poljubni dve točki (x_1, y_1) in (x_2, y_2) lahko med njima potegnemo daljico. Tako lahko napišemo enačbo premice:

$$\begin{aligned} y_1 &= ax_1 + b \\ y_2 &= ax_2 + b \end{aligned} \tag{8}$$

Iz zgornje enačbe lahko izračunamo parametra a in b. Če postopek ponovimo za vse možne kombinacije točk dobimo tako množico parov $(a,\ b)$. Parameter a predstavlja tangens kota daljice ali naklon, b pa začetek črte. Če na parameteru a izvedemo operacijo $a' = \arctan a$ dobima znan razpon kotov $a' \in [-\pi/2, \pi/2]$. Tako imamo za oba parametra a' in b znan razpon. Tako lahko vsak par (*a', b*) zapišemo v matriko pogostosti, ter s tem dobimo polja, ki so najbolj pogosta. Tako lahko narišemo premice, ki najbolje popisujejo točke.

4.22 Polnjenje področja ali *Region filling*

Za črnobelo rastrsko datoteko je potrebno uporabiti rekurzivni razlivni algoritem ki bo zapolnil notranjost za izbrano začetno točko. Uporabi se lahko slikarjev algoritem[7] (Painter's algorithm).

4.23 Bresenhamov algoritem za črte in kroge z antialiasingom

Celoštevilčni algoritem za risanje črt in krogov [7, 3] omogoča hitro risanje na rastrske enote. Izdelati je potrebno program, ki v rastrsko datoteko zapiše kroge in črte s tem, da izvede še antialiasing glede na barvo podlage.

4.24 Vizualizator 3D točk iz zajema

V laboratoriju LECAD smo razvili primitivni mehanski digitalizator 3D točk s tremi segmenti in A/D konverterjem, ki se ga priklopi na paralelni port PC-ja. Razvit je program, ki digitalizira točke in jih transformira v kartezijeve koordinate.

Za digitralizirane podatke, je potrebno izdelati program, ki ajete točke na primeren način prikaže v prostoru in ji tudi zapiže v formatu primernem za uvoz v 3D modelirnik.

4.25 Relaksacija mreže z mehurčkovo metodo v 2D

Z pakiranjem krogov[8] v omejen lik v ravnini dobimo lepo urejeno trikotniško mrežo z najmanjšo energijo tlačenja krogov (mehurčkov). Izdelajte program, ki uporabi omenjeno teorijo in za podano mrežo in konstantno funkcijo gostote mehurčkov simulira gibanje in s tem razbremeni podano trikotniško mrežo.

4.26 Glajenje mreže v 3D

Enostavno glajenje mreže je možno z premikanjem vozlišč elementov v geometrično sredino sosednjih vozlišč. Nov položaj notranje točke določimo z

$$\vec{x}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} \vec{x}_{ij} \tag{9}$$

kjer je x_{ij} položaj n_i sosednjih vozlišč povezaniih na točko i. Proces konvergira v dveh do petih iteracijah. Problemi lahko nastanejo pri konkavnih površinah.

Izdelati je potrebno program, ki bo prebral mrežo popisano v vhodni datoteki in izrisal začetno in končno stanje.

4.27 Advancing front generiranje mreže

Za 2D območje popisano z množico točk lomljenke je potrebno izdelati program, ki bo postavljal trikotnike v notranjost z uporabo algoritma s postavljanjem naprej. Več podatkov o algoritmu je podano v Techincal Watch Report[14].

4.28 Kvaternioni

Interpolacija orientacije naj bi se izvedla indirektno z interpolacijo kotov in ne s transformacijo matrik. V dveh dimenzijah imamo samo en kot zato je vrtenje enostavno. V treh dimenzijah pa imamo tri kote in s tem vrtenje krogle. Interpolacija je zato bolj zahtevna za prikaz.

Definicija kvaterniona Kvaternioni so sestavljeni iz treh imaginarnih velčin i, j in k in skalarnega dela a

$$q = a + bi + cj + dk \tag{10}$$

Pravila za sestavljanje imaginarnih veličin so naslednja:

$$\begin{aligned} i^2 = j^2 = k^2 &= -1 \\ ij = -ji &= k \\ jk = -kj &= i \\ ki = -ik &= j \end{aligned} \tag{11}$$

Lastnosti kvaternionov Produkt kvaternionov ni komutativen. Kvaternioni se lahko razdelijo na skalarni in vektorski del:

$$q = (s, \vec{v}) = s + v_1 i + v_2 j + v_3 k \tag{12}$$

V vektorski notaciji lahko zapišemo produkt dvek kvaternionov kot:

$$q_1 q_2 = (s_1 s_2 + \vec{v}_1 \cdot \vec{v}_2, s_1 \vec{v}_2 + s_2 \vec{v}_1 + \vec{v}_1 \times \vec{v}_2) \tag{13}$$

[14] http://www.epcc.ed.ac.uk/epcc-tec/documents/tw-meshgen/

Enotski kvaternion

- Enotski kvaternion ima enotsko normo
- Je izomorfen na orientacijo

Enotski kvaternion podan z

$$q_r = (\cos(\theta), \sin(\theta)\vec{v}) \tag{14}$$

za enotski vektor $\vec{v}$ je ekvivalenten rotaviji za 2θ okoli osi definirani z $\vec{v}$. q_r je ekvivalenten $-q_r$ ko ga interpretiramo kot orientacijo.

Kvaternioni imajo tudi konjugate: $\bar{q} = (s, -\vec{v})$. Norma kvaterniona je definirana z

$$|(s, \vec{v})| = \sqrt{s^2 + v_1^2 + v_2^2 + v_3^2} \tag{15}$$

Rotacija in kvaternioni Točke v prostoru so lahko predstavljene s kvaternioni, če imajo nični skalarni del. Točka Pje predstavljena kot:

$$q_p = (0, P) \tag{16}$$

Rotacija točke P je definirana z q_r

$$q'_P = q_r q_P q_r^{-1} \tag{17}$$

Za enotske kvaternione je inverz enak konjugatu.

Prednosti kvaternionov

1. Za veliko aplikacij je neodvisna definicija osi zavrtitve in kota primerna.
2. Definicija orientacije s kvaternionom je neodvisna od koordinatnega sistema in izotropna.
3. Krožna interpolacija kvaternionov daje boljše rezultate kot Eulerjevi koti.

Interpolacija s kvaternioni

- Linearna interpolacija.
  Za interpolacijo dveh orientacij lahko uporabimo linearno interpolacijo vseh veličin (ga smatramo kot štiridimenzionalni vektor). Žal pa vmesni kvaternioni ne bi imeli enotske norme. Kotna hitrost ne bi bila konstantna tudi če bi renormalizirali.
- Krožna interpolacija.
  Za interpolacijo dveh orientacij lahk ouporabimo krožno interpolacijo tako, da uporabimo kvaternione kot vektorje.
  Kot med dvema kvaternionoma je

  $$\omega = \arccos(q_1 \cdot q_2) \tag{18}$$

  Za dani parameter $u = \in [0, 1]$ je kvaternion zavrtitve s krožno interpolacijo podan kot

  $$q(u) = q_1 \frac{\sin((1-u)\omega)}{\sin(\omega)} + q_2 \frac{\sin(u\omega)}{\sin(\omega)} \tag{19}$$

Naloga. Izdelati je potrebno program, ki bo mrežni model popisan v prostoru zavrtel iz enega položaja v drugi z uporabo krožne interpolacije s kvaternioni. Potrebno je rešiti[15] tudi problem singularnosti pri $\omega \approx 0$ in $\omega \approx n\pi/2$. Izris 3D modela in vmesnih stopenj z uporabo knjižnice PHIGS ali OPEN GL.

[15]Lahko z linearno interpolacijo.

4.29 Vrtenine

Veliko objekov v CAD modelirnikih je narejeno z zavrtitvijo krivulje okoli osi. Tako je todoid narejen z zavrtitvijo kroga okoli osi, ki je v ravnini kroga. Izdelati je potrebno program, ki bo prebral vhodno datoteko s podatki o vhodni krivulji, osi in kotu vrtenja, jo prikazal v poljubni orientaciji, ter zapisal v DXF ali Inventor datoteki. Podreben vhodni podatek, je tudi število krožnih delitev.

4.30 Raztegnjene ploskve

Podobno kot pri vrteninah imamo pri zavesnih površinah dve tvorni krivulji:

- osnovna krivulja, ki jo raztezamo
- krivulja raztega

Zahteve za program so enake kot pri 4.29.

4.31 Osmiška drevesa

So tridimenzoinalne podatkovne strukture [10] s katerimi razdelimo prostor na urejeno množico podprostorov, ki so razdeljena glede na zasedbo objektov v prostoru. Štiriška drevesa so dvodimenzionalni ekvivalent osmiških dreves. Izdelati je potrebno program, ki bo za podane krogle in cilindre izvedel razdelitev prostora in jo prikazal.

4.32 Senčenje s tehniko Gouraud

Tehnika inkrementalnega senčenja se uporablja za hiter prikaz v različnih programih za 3D modeliranje. Nekateri računalniki imajo to tehniko vgrajeno že v grafičnem pospeševalniku.

Za podan ravninski lik v prostoru se na vozliščih izračuna svetlost (intenziteta). Svetlost v notranjosti lika pa se določi z bilinearno interpolacijo vozlišč.

Za podano vhodno datoteko trikotnikov in pogledom v prostoru, je potrebno zapisati rastrsko datoteko z rasteriziranimi trikotniki.

4.33 Senčenje s tehniko Phong

Podobno kot v 4.32 je Phongova inkrementalna tehnika za hitro senčenje. Je pa počasnejša od Gouraud, ki jo lahko naredimo tudi s celoštevilčnimi rutinami. Phongova metoda je interpolacija normal in šele nato določitev intenzitete.

Za vhodno datoteko v kateri je podana mreža $n \times m$ 3D točk s položajem liči in pogleda je potrebno izdelati rezultat Phong senčenja v rastrski datoteki. Normale v vozličih se do ločijo kot normiran seštevek sosednjih ploskvic.

4.34 NURBS

Izdelati je potrebno rekurziven program, ki bo prikazal 2D krivulje NURBS in izbrane primere z razlago.

4.35 Krivulje beta

Podobno kot za krivulje NURBS je potrebno prikazati krivulje β. Podrobnejša razlaga teh krivulj, ki vnašajo dodatne parametre za nadzor poteka so podane v knjigi *Computer graphics and geometric modeling using Beta-splines* avtorja Braina Barskija.[2]

5 Informacije

Podobne vaje se lahko določi tudi na govorilnih vajah. Zahtevnost vaj je koncipirana na predvideno število ur. Govorilne vaje: vsak dan od 12-13 v laboratoriju LECAD ali po dogovoru s tutorjem posamezne naloge. Potrebna navodila se nahajajo tudi na WWW strežniku laboratorija LECAD[16]. Možna je tudi komunikacija z elektronsko pošto s tutorji nalog na naslov `rpk@lecad.uni-lj.si`.

Literatura

[1] A.H. Barr. *Rigid Phisically based superquadrics*, chapter III.8, pages 137–159. Number III in Graphics Gems. Academic Press, 1992.

[2] Brian Barsky. *Computer graphics and geometric modeling using Beta-splines*. Springer-Verlag, 1988. CTK 45714/2.

[3] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Huges. *Computer Graphics*. Addison Wesley, 2 edition, 1990. CTK 207495 INF-E.

[4] Paul W. Heiser. *Salvaging damaged dbase files*. Comtech Publishing, 1987.

[5] Ketan Mulmuley. *Computational Algorithms*. Prentice Hall, 1994.

[6] Kenton F. Musgrave, Darwyn Peachet, Ken Perlin, and Steven Worley. *Texturing and Modeling*. Academic Press, 1994.

[7] Roy A. Plastock and Gordom Kalley. *Theory and problems of computer graphics*. Schaum outline series. McGraw-Hill, 1986.

[8] Kenji Shimada and David C. Gossard. Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing. In *Third Symp. on Solid Modeling and Appls.*, pages 409–419, May 1995. http://www.trl.ibm.co.jp/projects/s7340/meshing/meshingE.htm.

[9] Robert Ulichney. *Digital Halftoning*. The MIT Press, 1990.

[10] Alan Watt. *3D Computer Graphics principles and practice*. Addison Wesley, 2 edition, 1993. CTK 203797 INF-B.

[16] http://www.lecad.uni-lj.si/documents/