รายการอ้างอิง

ภาษาไทย

1. ธีรวัฒน์ ประกอบผล. การพัฒนาระบบเก็บข้อมูลด้วยเทคนิคโทรทัศน์สำหรับคำนวณสร้างภาพโทโมกราฟี. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2537.
2. มงคล วรรณประภา. การพัฒนาระบบสแกนด้วยรังสีแกมมาเพื่อคำนวณการสร้างภาพโทโมกราฟีของเสาคอนกรีตเสริมเหล็กวิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2536.
3. มานัส มงคลสุข. พื้นฐานทางฟิสิกส์ของCT และ MRI. กรุงเทพมหานคร : ไพศาลศิลป์การพิมพ์, 2532.
4. วัลยา เอี่ยมสุรนันท์. การพัฒนาโปรแกรมสำเร็จรูปแบบคำนวณสร้างภาพโทโมกราฟีเพื่อการตรวจสอบชิ้นงานอุตสาหกรรม. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2538.
5. สมยศ ศรีสถิตย์, อรรถพร ภัทรสุมันต์. การคำนวณสร้างภาพโทโมกราฟีด้วยเทคนิคฟิล์มเพื่อการตรวจสอบแบบไม่ทำลาย. โครงการวิจัย เลขที่ 134-MRD-2538 ทุนส่งเสริมการวิจัยวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2538.

ภาษาอังกฤษ

1. Anll K. Jain. Fundamentals of digital image processing. Prentice-Hall international editions, 1989.
2. Lee Adams. Lee Adams' visualization graphics in C. Windcrest Books, 1991.
3. Steve Rimmer. Bit-mapped graphics. Windcrest Books, 1990.

ภาคผนวก

ภาคผนวก ก

PCX

Summary: PCX

Image type Bitmap (raster)

Intended use Proprietary format for PC-based paint program, now quite public

Owner Zsoft Corporation. 450 Franklin Rd. Suite 100, Marietta GA 30067, (404) 428-0008.

Contact: code librarian Dean Ansley

Latest revision/version/release date 1991

Platforms PC

Supporting applications Desktop publishing, graphics arts, video capture

Similar to Somewhat like MacPaint in compression, but capable of far superior image quality

Overall assessment Commonly supported, therefore good for exchange in the PC environment; reasonably efficient storage for paint-type images (i.e., images with large areas of constant tone); tricky to interpret because of the many possible variations.

Advantages and disadvantages

Advantages PCX is one of the oldest, and therefore most widely-supported bitmap for-mats for PCs. Current versions support 24-bit color, implemented either as a palette of up to 256 colors or full 24-bit RGB, with image sizes up to 64K x 64K pixels. Data are compressed through run-length encoding. Zsoft offers a technical bulleting on PCX and also provides function call information on FRIEZE, its PCX-writing screen-capture program, for application programmers.

Disadvantages The file format has no provisions for storing greyscale or color correction table; neither does it store CMYK- or HIS-model data (although some Zsoft programs allow you to adjust color values using HIS). Its run-length compression scheme can be inefficient, especially for scanned or video images. PCX files can use a variety of palette techniques due to the age and evolution of PCX, as a result of which most readers cannot handle the full range of possible PCX implementations.

Variants

PCX is upgraded with new versions of Zsoft products; a code in the file header determines what version of Zsoft products the file supports. The variations, listed by header code, are given in Tables A-1 and A-2.

Table A-1 Zsoft products corresponding to pcx version number

Version number	Zsoft product and version
0	Version 2.5 of PC Paintbrush
2	Version 2.8 of PC Paintbrush with palette information
3	Version 2.8 of PC Paintbrush without palette information
4	PC Paintbrush for Windows (Not the "plus" version)
5	Version 3.0 and higher of PC Paintbrush and PC Paintbrush IV Plus; also PC Paintbrush Plus for Windows; Publisher's Paintbrush

Table A-2 Image characteristics corresponding to PCX version number

Version number	Image characteristics
0	Basic monochrome (2-color) or 4-color image
2	As above, plus 16-color images
3	As above, plus 256 colors from 24-bit palette; also full 24-bit RGB color

These variations make more sense if you realize that they roughly track the growth in graphics capability of the PC, from the CGA (Color Graphics Adapter), to the EGA (Enhanced Graphics Adapter) and VGA (Video Graphics Adapter), and finally, the extended VGA and beyond.

The version number is not, however, a guide to how the image is encoded. A file with a version code of 5, for instance, might employ a palette or not. If you are creating a PCX reader, the most reliable guideline to decoding comes from the "bits per pixel" and " number of color planes" information in the header, as discussed below under " Keys to interpreting data."

Overview of file structure

The PCX format is comprised of three parts: the file header, bitmap data, and (in later versions) a palette of up to 256 colors.

The file begins with a header of a fixed 128 bytes. It contains, in addition to the version number, resolution of the printed or scanned image (in dots per inch), dimensions (in pixels), bytes per scan line, bits per pixel, and the number of color planes. The header might also contain a palette, and a code indicating whether that palette is grayscale or color.

The heart of the file is bitmap data. Bitmap data are recoded in run-length compressed form similar to the Packbits compression scheme. The pixel values are generally one-byte pointers to locations in a palette.

If the version code is 5, and thereis a single bit plane, there is a 256-color palette of RGB values, one byte for each primary (R, G, and B), at the end of the file.

Format details

The PCX format is relatively simple to write, but tricky to read unless the details of the encodeed image (such as bit depth and palette) are already known. Accordingly, the following description focuses on the worst case: reading a PCX file of indeterminate characteristics and age. All number are in little-endian (Intel) format.

Header

The header is summarized by the following table.

Table A-3 PCX file header

Start Byte	Size in Bytes	Contents	Interpretation
0	1	Zsoft flag	10(0A hex) = Zsoft PCX file
1	1	Version no.	0 = PC Paintbrush 2.5
			2 = PC Paintbrush 2.8 with palette
			3 = PC Paintbrush 2.8 without palette
			4 = PC Paintbrush for Windows
			5 = PC paintbrush 3.0 and up; also PC Paintbrush IV, IV Plus; Publisher's Paintbrush
2	1	Encoding	1 = PCX run-length encoding
3	1	Bits per pixel	Number of bits/pixel, each plane
4	8	Image dimensions	Image limits as Xmin, Ymin, Xmax, Ymax in pixel units
12	2	Horiz. Resolution	Dots/inch in X, when printed
14	2	Vert. Resolution	Dots/inch in Y, when printed
16	48	Header palette	See palette discussion
64	1	Reserved	always 0
65	1	Planes	Number of color/ grayscale planes
66	2	Byts per line	Memory needed for one color plane of each horiz. Line
68	2	Header palette interp.	1 = color or B&W
			2 = grayscale
70	2	Video screen size, X	Number of pixels horiz. of video output
72	2	Video screen size, Y	Number of pixels vert. Of video output
(74	54	Blanks to end of header)	

Byte 0, Zsoft flag Always decimal 160, hex A0.

Byte 1, Version number A somewhat unreliable guide to what the file contains; see the preceding discussion, " Variations. "

Byte 2, Encoding As of this writing, always 1. There is currently only one " encoding " (compression) method, the run-length scheme described below in " Bitmap data."

Byte 3, Bits per pixel Actually, bits per pixel per plane. Might take the value 1, 2, 4, or 8.

Bytes 4-11, Image dimensions The image dimensions are given as minimum and maximum limits, even though a single set of dimensions would suffice. Generally the minimum limits are zero. All limits appear as 16-bit unsigned intergers, in units of pixels, as noted. Image size in pixels can be computed by: XSIZE = Xmax - Xmin + 1; YSIZE = Ymax - Ymin + 1.

Bytes12 - 15, Horizontal and vertical resolution in dots per inch These two 16-bit numbers are somewhat extraneous. They play no part in defining the stored image; however, when combined with the image dimensions, they communicate the original size of a scanned image, or intended size of a printed image, in inches.

Bytes 16 - 63, Header palette This field appears to be used only for files with a single bit plane, 16 or fewer colors, and a version code of 2. (See " Keys to interpreting data, " following.) When used, the palette holds 16 " triples " of 1-byte palette values. See the following section for details.

Byte 65, Planes PCX images can be a single plane or multiple color planes (see chapter 1). This header byte gives the number of planes, and is critical to proper interpretation of the PCX file.

Byte 66, Bytes per line Actually, bytes per line per plane - the number of bytes of memory required to store one plane of one scan line of the unpacked image; always an even number.

Byte 68, Header palette interpretation 1 = color/monochrome; 2 = grayscale. Not used in Paintbrush IV or IV Plus.

Bytes 70 - 73, Vide0 screen size, X and Y Used only by Paintbrush IV and Paintbrush IV Plus; not essential, but possibly useful for communicating proper aspect ratio (avoiding squeeze-type distortion).

Keys to interpreting data

Because data can recorded in several ways in a PCX file, there must be reliable indicators of those ways. As mentioned previously, the version number is not an adequate guide.

The most reliable guide are the bits per pixel (header byte 3) and number of color planes (header byte 65). Table A-4 shows how data should be interpreted according to these guides.

Table A-4 Interpretation of PCX data

Bits/pixel/plane	**Number of planes**	**Interpretation**
1	1	monochrome
1	2	4-color
1	3	8-color
1	4	16-color
2	1	4-color,CGA header palette*
2	4	16-color
4	1	16-color, EGA header palette*
8	1	256-color, palette at file's end
8	3	16.7M-color

*Note: In asterisked modes, the header palette is used unless the version code is 5 or greater, in which case the palette at the end of the file is used.

The number of planes tells whether or not a palette was used. More than one plane means " no palette. " If a palette is used, a combination of version number and bits per pixel determines which of the several kinds of palette supported by PCX is used.

Bitmap data

If no palette is used, the data are actual pixel values; otherwise, they are pointers to palette values. In the latter case, they give the offset (in 3-byte triplets, e.g., 1 = byte 3) from the start of whatever palette is used.

When data are actual pixel values, they are stored by scan line, by color plane. For instance, for three colors, red, green, and blue (RGB), the data are formatted:

(Line 0:) RRRRRR...GGGGGG...BBBBBB... (Line 1:) RRRRRR...GGGGGG...BBBBBB...

If there are two planes, the colors are arbitrary. If there are three planes planes, the colors are RGB. When four color planes are used, they are the 1-bit planes of the IBM CGA/EGA standard:

red, green, blue, and " intensity " (RGBI). The intensity bit simply gives a pixel a nominally higher brightness.

When data are pointers to a palette, they comprise a complete image plane (i.e., they are not broken up into separate color planes). The data are simply formatted as follows (where the P characters represent various pointer values):

(Line 0:) PPPPPPP (Line 1:) PPPPPPP...

The length of P depends on the pixel depth in bits/pixel/plane; for instance, with a deapth of four bits, P is half of a byte.

In all cases, there are coding breaks between scan lines. There are, however, no coding breaks between color planes within a scan line. There is also no delimiting symbol provided to mark the end of a scan line (although a scan line might or might not end with extra zeros). That is, there are no line numbers like those shown here, or null characters, spaces, carriage returns, linefeeds, or other characters between scan lines.

No matter what type of bitmap data is recorded, the same run-length compression scheme is used. The unpacking algorithm is as follows. (In the palette-based images, which are the kind most commonly used today, there is only one plane.)

Unpacking algorithm.

```
Read Xmax, Xmin, Ymax, Ymin, from header, bytes 4-11.
Compute image size (number of pixels) from header data: (Xmax - Xmin + 1) x (Ymax - Ymin + 1)
Read number of planes from header, byte 65.
Read bytes per line (per plan) from header, byte 66.
For each scan line,
    For each plane of the scan line:
        Read a byte.
        Are top two bits = 1?
            If so, the next 6 bits of the byte is a run-length count, N
                The next byte holds data; read it, then duplicate it N times.
            If not, the byte itself is a single data value.
```

When total bytes recovered exceeds the " bytes per line " number that was read from the header, reset byte counter and begin the next plane.

When all planes have been recovered, begin the next scan line.

When total bytes recovered (all lines, all plane) exceeds the computed image size, quit.

Trim image at right-hand edge to conform to computed X dimension in pixels.

Note the limits of this compression scheme. Data may not exceed eight bits per plane. Furthermore, run length is limited to a maximum of by the six-bit run-length count. Also note that because data is compressed on byte boundaries, and because Bytes Per Scan Line must be even, there will often be extra at the end of each scan line.

Palettes

As indicated by Table - above, any PCX file that has more than one bit per pixel, yet has only one color plane, uses a palette. Unfortunately, the evolution of graphics adapters on the PC has resulted in three different palette implementations in PCX, and it requires careful examination to figure out which implementation is to be used. Files with a version code of five are the easiest to ascertain. If they have a single color plane, they use the -color "VGA" palette at the end of the file. Other palette-based files use the header palette, which has two possible implementation: EGA and CGA

256-color, end-of-file "VGA" palette The 256-color palette starts 768 bytes befor the end of file (EOF), and is preceded by a code of 12 decimal ()C hex). (768 = 256 x 1 byte for each of R, G, and B). A pixel value of n, therefore, points to location EOF-768+3*n in the palette; the next 3 bytes contain the red, green, and blue values, respectively, for that pixel. However, because the VGA only provides 64 levels for R, G, and B, the palette values must be divided by four when reading a PCX file for VGA display

16-color " EGA/VGA " header palette The header palette, located at bytes 16-63, is organized into triples: sixteen sets of three data bytes, one byte each for R, G, and B. In files created with (or for) an EGA, each primary color can take only four level, so the ponds to a level: 0-63 = level 0, 64-127 = level 1, 128 -192 = level 2, and 193-254 = level 3.

"CGA" header palette (This palette is considered obsolete, and not used for files with a version code of five or higher.) In this scheme, only the top-end bits of bytes 16 and 19 are needed (header offset 0 and 3).

Byte 16 The top nibble (4 bits) of byte 16 provides a value between 0 and 15 called the background color. Background color is used as follows. The CGA supports two alternative " 4-color " palette. (Header byte 19 determines which is used, as described in the following paragraph.) The first (Palette 0, or the " yellow " palette) is comprised of background color, plus green, red, and yellow (brown on some CGAs). The second (Palette 1, indicates that this color is variable; it can actually take on any of 16 values generated by the 4-bit IRGB code given in the top nibble of byte 16.

Byte 19 The top three bits of byte 19 give the variables C, P, and I. The C stands for color burst enable, when 0 = color and 1 = monochrome. The P stands for palette, where 0 = " yellow " palette and 1 = " white " palette. And I stand for intensity, where 0 = dim and 1 = bright (a variation on the two alternative CGA palettes).

24 -bits color

Some recent Zsoft products produce PCX files with full 24-bit color (no palette). These can be identified by a version code of five or higher, with eight bits/pixel/plane and three-bit planes.

ภาคผนวก ข

Source code โปรแกรมการคำนวณสร้างภาพโทโมกราฟีแบบสามมิติ

- CT_3D.C
- CT.H
- EDIT.H
- GR3D.H

SOURCE CODE OF CT_3D.C

```c
#include "include.h"


#include "BkDefs.H"
#include "BkMath.H"
#include "BkGraph.H"
#include "BkGlobs.h"



#define OK      1
#define OVERFLOW -1
#define MAX     30
#define EVEN0    0

#define K_ESC    0x11b
#define K_TAB    0xf09
#define K_UP     0x4800
#define K_DOWN   0x5000
#define K_LEFT   0x4b00
#define K_RIGHT  0x4d00
#define K_ENTER  0x1c0d
#define K_PLUS   0x4e2b
#define ALT_F1   0x6800
#define CTR_F1   0x5e00
#define K_F1     0x3b00
#define K_F2     0x3c00
#define K_F3     0x3d00
#define K_F4     0x3e00
#define K_F5     0x3f00
#define K_F6     0x4000
#define K_F7     0x4100
#define K_F8     0x4200
#define K_F9     0x4300
#define K_F10    0x4400
#define CTR_HOME 0x7700
#define K_HOME   0x4700
#define K_END    0x4F00
/*
#define K_F11
#define K_F12
*/
#define RIGHT 0x01
#define LEFT  0x02
#define CTRL  0x04
#define ALT   0x08
#define BACKSPACE 0xE08
#define SPACE  0x3920

time_t getEVEN(unsigned  *even);
void DOCOMMAND(char *s);
void SaveData(void);

char ProfilePath[80];
char SectionPath[80];
char NProfile[70][14];
char NSection[70][14];
char NConfig[255];
int No_Profile,No_Raysum,No_Section;
float AngularStep,Ra_Interval,Sec_Interval;
int PicLeft,PicTop,PicRight,PicBottom;
int PicMaxX,PicMaxY;
int PicTC,PicLC;

float minAll = 1000,maxAll = -1000;
float minSel = 1000,maxSel = -1000;
int ColMode = 2;
int SorMode = 2;
int IDENPASS = 0;
int PLAN = 0;

float Deviation = 1;
float Air_Value = 0;
float Interest = 0;
char AirStatus = 0;
char PIBS = 0;
char RealOb = 1;
int  FormFile = 0;

#include "ct.h"
#include "tools.h"
#include "d640_16.h"
#include "select.h"

struct POSITION {
                              int x;
                              int y;
                           };
struct G_WINDOW {
                              int Left;
                              int Top;
                              int Right;
                              int Bottom;
                           };

long    STACK_KEEP[MAX];
long    QUEUE_KEEP[MAX];
int     STACK_INDEX = 0;
int     QUEUE_INDEX = 0;

struct POSITION CurPosition;

time_t getEVEN(unsigned  *even)
{
unsigned  item;
   if( (get_KB_BUFFER(&item)) == OK)  *even =
item;
                 else *even = EVEN0;
                 CurPosition.y = wherey();
                 CurPosition.x = wherex();
                 return time(NULL);
}

int get_KB_BUFFER(int *item)
{
int ret;
   if ( (kbhit()) == 0 ) ret = !OK;
   else  {  *item = bioskey(0); ret = OK; }
   return ret;
}

void clr_KB_BUFFER(void)
{
```

```
while(1)
 {
  if ( (kbhit()) != 0)
     bioskey(0);
  else
               break;
 }
}
int QUEUE_PUSH(long item)
{
int ret;
  if(QUEUE_INDEX >= MAX)
   {
    ret = !OK;
               /* QUEUE OVER FLOW */
   }else
   {
    QUEUE_KEEP[QUEUE_INDEX] = item;
    QUEUE_INDEX++;
    ret = OK;
    /* ADD QUEUE OK */
   }
  return ret;
}
int QUEUE_POP(long *item)
{
int ret;
int i;
  if(QUEUE_INDEX < 0)
   {
    ret = !OK; /* QUEUE EMTRY */
   }else
   {
    *item = QUEUE_KEEP[0];
               for(i = 1;i<QUEUE_INDEX;i++)
QUEUE_KEEP[i-1] = QUEUE_KEEP[i];
    QUEUE_INDEX--;
    ret = OK; /* POP UP OK */
   }
  return ret;
}
int STACK_POP(long *item)
{
int ret;
  if(STACK_INDEX < 0)
   {
               ret = !OK; /* STACK EMTRY */
   }else
   {
    STACK_INDEX--;
    *item = STACK_KEEP[STACK_INDEX];
    ret = OK; /* POP UP OK */
   }
  return ret;
}
int STACK_PUSH(long item)
{
int ret;
  if(STACK_INDEX >= MAX)
   {
               ret = !OK;
    /* STACK OVER FLOW */
   }else
   {
```

```
    STACK_KEEP[STACK_INDEX] = item;
    STACK_INDEX++;
    ret = OK;
    /* ADD OK */
   }
  return ret;
}


char buffer[4096];


void DOCOMMAND(char *s)
{
char Command[80];
char Arg[80];
int i;
int j = 0;
int f = 0;
               Arg[0] = '\0';
               for(i=0;i<80;i++)
               {
                              switch(s[i]){
                              case ' ': break;
                              case '\0':
                              case '\n':
               if (f == 0) Command[j] = '\0';
               else Arg[j] = '\0';
               i = 80; break;
                              case '=':
               Command[j] = '\0';
               f = 1; j = 0; break;
                              case ';':
               if (f == 0)     Command[j] = '\0';
               else Arg[j] = '\0';
               i = 80; break;
                              default:
               if (f == 0) Command[j] = s[i];
```

```
                else Arg[j] = s[i];


                j++; break;

                                }
                }
                strlwr(Command);
                strlwr(Arg);
/* ——— */
                if ( !strcmp(Command,"vdomode") )
{ f = 1;goto AAA; };
                if ( !strcmp(Command,"targetfile") )
{f=2; goto AAA;};
                if ( !strcmp(Command,"sourcefile") )
{f=3; goto AAA;};
                if (
!strcmp(Command,"no.ofprojection"))  {f=4;goto
AAA;};
                if ( !strcmp(Command,"projections"))
{f=4;goto AAA;};
                if ( !strcmp(Command,"no.ofraysum")
) {f=5;goto AAA;};
                if ( !strcmp(Command,"ray-sum") )
{f=5;goto AAA;};
                if ( !strcmp(Command,"ray-
suminterval") ) {f=6;goto AAA;};
                if ( !strcmp(Command,"angularstep")
) {f=7; goto AAA;};
                if (
!strcmp(Command,"minimumvalue") ) {f=8; goto
AAA;};
                if (
!strcmp(Command,"maximumvalue") ) {f=9; goto
AAA;};
                if (
!strcmp(Command,"graphicfile")){f=10;goto AAA;};
                if (
!strcmp(Command,"showgraphic")){f=11;goto AAA;};
                if ( !strcmp(Command,"readct.ini") )
{f=12;goto AAA;};
                if (
!strcmp(Command,"reconstruction") ) {f=13;goto
AAA;};
                if ( !strcmp(Command,"cls") )
{f=14;goto AAA;};
                if ( !strcmp(Command,"test") )
{f=15;goto AAA;};
                if ( !strcmp(Command,"ctfile"))
{f=16;goto AAA;};
                if ( !strcmp(Command,"quit"))
{f=17;goto AAA;};
                if ( !strcmp(Command,"draw16"))
{f=18; goto AAA;};
                if ( strlen(Command) )            f =
1000; else f =0;
AAA:
                switch(f){

case 0:  break;

case 1:  Mode_vdo = atoi(Arg);  break;
```

```
case 2:


strcpy(targetFile,Arg); break;

case 3:  strcpy(sourceFile,Arg); break;

case 4:

                                        pjNO = atoi(Arg);
break;

case 5:

                                        rsNO = atoi(Arg);
break;

case 6:

                                        rayInterval =
atof(Arg); break;

case 7:

                                        angularStep =
atof(Arg);  break;

case 8:

                                        min_value =
atof(Arg); break;

case 9:

                                        max_value =
atof(Arg); break;

case 10:


strcpy(targetFile,Arg); break;

case 11:   gettext(1,1,80,25,buffer);


                        graphDraw();


                        puttext(1,1,80,25,buffer);


                        gotoxy(0,CurPosition.y+1);


                        break;

case 12: readINI(); break;

case 13: com_tar(1); break;

case 14: clrscr(); break;

case 15:/* showpoint(); */
case 16:  strcpy(otFile,Arg); break;

case 17: exit(0);

case 18:   gettext(1,1,80,25,buffer);


                VGA16graph();


                drawVGA16();


                closegraph();


                puttext(1,1,80,25,buffer);


                gotoxy(0,CurPosition.y+1);


                break;

default:

                                printf("\r-->
COMMAND NOT FOUND %s\n",Command);
                                                        };
}


void block(int left,int top,int right,int bottom)
{
                line(left,top,right,top);
                line(left,top,left,bottom);
                line(right,top,right,bottom);
                line(left,bottom,right,bottom);
}




void DisZFile(void)
{
int i,j,q = 1;
char tmp[20];


do{
clrscr();
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqw\n");
for(i=0;i<23;i++)
printf("x
x\n");
printf("mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqq");
   for(i=0;i<No_Section;i++)
   {
                if(i<20){
gotoxy(5,2+i);printf("%2i - %s",i,NSection[i]);
                                }
                if((i>19)&&(i<40)){

gotoxy(35,i-18);printf("%i - %s",i,NSection[i]);
                                        }
                if((i>39)&&(i<60)){

gotoxy(62,i-38);printf("%i - %s",i,NSection[i]);
                                        }
  }
gotoxy(2,22);
printf("<q> to quit-----------------------------
------------------");
gotoxy(30,23);printf("SELECT NUMBER ->
");scanf("%s",tmp);

if((tmp[0] != 'q') && (tmp[0] != 'Q')) {

                                j = atoi(tmp);


                strcpy(targetFile,NSection[j]);

                                drawVGA256();

                                }else {q = !q;}
}while(q);
}




#include "edit6.h"
#include "gr3d.h"



void SaveData(void)
{
FILE *wrt;
int i;
char tmp[14];
char *p;
   IDENPASS = 1;
   strcpy(tmp,NSection[0]);
   p =  strtok(tmp,"0");
   if(p)    sprintf(NConfig,"%s.IDF",p);
   else  { p = strtok(tmp,".");
sprintf(NConfig,"%s.CON",p);}
if ((wrt = fopen(NConfig, "wt"))    == NULL)
{
  fprintf(stderr, "Cannot open input file.\n");
  exit(0);
}
  fprintf(wrt,"No. of Projection = %f\n",No_Profile);
fprintf(wrt,"No. of Raysum = %f\n",No_Raysum);
  fprintf(wrt,"No. of Section = %f\n",No_Section);
fprintf(wrt,"Angularstep = %f\n",AngularStep);
  fprintf(wrt,"Ray-sum Interval = %f\n",Rs_Interval);
fprintf(wrt,"Section Interval = %f\n",Sec_Interval);
```

```
  fprintf(wrt,"Picture Max X = %f\n",PicMaxX);
fprintf(wrt,"Picture Max Y = %f\n",PicMaxY);
  fprintf(wrt,"Picture Left = %i\n",PicLeft);
fprintf(wrt,"Picture Top = %i\n",PicTop);
  fprintf(wrt,"Picture Right = %i\n",PicRight);
fprintf(wrt,"Picture Bottom = %f\n",PicBottom);
  fprintf(wrt,"Minimum All = %f\n",minAll);
fprintf(wrt,"Maximum All = %f\n",maxAll);
  fprintf(wrt,"-SOURCE FILE-\n");
  for(i=0;i<No_Profile;i++)
fprintf(wrt,"%s\n",NProfile[i]);
  fprintf(wrt,"-SECTION FILE-\n");
  for(i=0;i<No_Section;i++)
fprintf(wrt,"%s\n",NSection[i]);
  fprintf(wrt,"-END DATA PROFILE-\n");
fclose(wrt);
}
void InputPro(void)
{
int indxS=-1;
int indxT=-1;
int i;
int quit=1;
char NNN[4096];
char Temp[25];
time_t st,fn,cu;
unsigned even;
clrscr();
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqฆ\n");
for(i=0;i<23;i++)
printf("ฆ
ฆ\n");
printf("ñqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqq");
gotoxy(27,2);printf("PROFILES DATA FROM
FILES");
gotoxy(2,3);printf("------------------------------------
-----------------------------");
//gotoxy(2,4) ;printf("NO. of Data Files :            ; ");
//gotoxy(40,4);printf("Angular Step      :            ; ");
//gotoxy(2,5) ;printf("NO. of ray-sum    :            ;
");
//gotoxy(40,5);printf("Ray-sum interval  :            ;
");
gotoxy(2,6) ;printf("No. of Data Files :            ; ");
gotoxy(40,6);printf("Section Interval  :            ; ");
gotoxy(2,7) ;printf("------------------------------------
-----------------------------");
gotoxy(2,8); printf("PROJECTIONS FILE
NAME(SOURCE FILE)");
gotoxy(51,8);printf("IMAGE FILE NAME(TARGET
FILE)");
gotoxy(2,9);printf("------------------------------------
-----------------------------");
gotoxy(40,9);printf("ข");
for(i=10;i<23;i++){gotoxy(40,i); printf("ฆ");};
gotoxy(2,23);printf("------------------------------------
-----------------------------");
gotoxy(40,23); printf("ง");
gotoxy(2,24);printf("<ESC> return MAINMENU ");
gotoxy(30,24);printf("<F-9> to edit");
```

```
gotoxy(54,24);printf("<F-10> to continue");
gotoxy(23,6);
do{
  cu = getEVEN(&even);
  switch(even){

                    case K_LEFT:
if((CurPosition.x>40)&&(CurPosition.y<15))
gotoxy(23,CurPosition.y);

if((CurPosition.x<40) && (CurPosition.y>4)
&&(CurPosition.y<15))

gotoxy(61,CurPosition.y-1);

if((CurPosition.x>40)&&(CurPosition.y>15))

gotoxy(5,20);

                  break;
                    case K_RIGHT:
if(CurPosition.x<40)
                  gotoxy(61,CurPosition.y);

if((CurPosition.x>40) &&(CurPosition.y <6) )

gotoxy(23,CurPosition.y+1);

if((CurPosition.x<40)&&(CurPosition.y>15))

gotoxy(45,20);

                  break;
                    case K_UP: if((CurPosition.y >
4)&&(CurPosition.y<15))

gotoxy(CurPosition.x,CurPosition.y-1);

                  if((CurPosition.y > 15) &&
(CurPosition.x>40))

gotoxy(61,6);

                  if((CurPosition.y>15) &&
(CurPosition.x<40))

gotoxy(23,6);
                                        break;
                    case K_DOWN: if((CurPosition.y
<6))

gotoxy(CurPosition.x,CurPosition.y+1);

if(CurPosition.y >=6) gotoxy(5,20);
                                        break;
                    case K_ENTER:

                  if((CurPosition.x<40)&&(CurPosition.
y>15))
                                             {
indxS++;gettext(5,20,29,20,NNN);

gotoxy(2,20);printf("%2i",indxS+1);
movetext(2,11,39,21,2,10); gotoxy(5,20);

for(i=0;i<12;i++)

{NProfile[indxS][i] = NNN[i*2];
                                        }

NProfile[indxS][13] = '\0';
                                       }

            if((CurPosition.x>40)&&(CurPosition.
y>15))
                                       {
indxT++;gettext(45,20,69,20,NNN);

gotoxy(42,20);printf("%2i",indxT+1);

movetext(42,11,78,21,42,10); gotoxy(45,20);

for(i=0;i<12;i++)

{NSection[indxT][i] = NNN[i*2];
                                        }

NSection[indxT][13] ='\0';
                                       }

            break;
              case
K_F10:gettext(22,4,37,4,NNN);

            for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

            No_Profile = atoi(Temp);

            gettext(22,5,37,5,NNN);

            for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

            No_Raysum = atoi(Temp);

            gettext(22,6,37,6,NNN);

            for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

            No_Section = atoi(Temp);

            gettext(60,4,75,4,NNN);

            for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

            AngularStep = atof(Temp);

            gettext(60,5,75,5,NNN);

            for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

            Ra_Interval = atof(Temp);
```

```
            gettext(60,6,75,6,NNN);

            for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

            Sec_Interval = atof(Temp);
//-

            if(indxS <= (No_Section))
                                       {

for(i=0;i<12;i++) {

                              if (NProfile[0][i]
!= ' ') Temp[i] = NProfile[0][i];

                              else {Temp[i] =
'\0';break;}

                              }

for(i=0;i<No_Section;i++)

            sprintf(NProfile[i],"%s%i.dat\0",Temp
,i);
                                       }

            if(indxT < (No_Section - 2))
                                       {

for(i=0;i<12;i++) {

                              if (NSection[0][i]
!= ' ') Temp[i] = NSection[0][i];

                              else {Temp[i] =
'\0';break;}

                              }

for(i=0;i<=No_Section+1;i++)

            sprintf(NSection[i],"%s%i.CTT",Temp
,i);
                                       }

//-

            quit=0;
//
            GrInput();

            SaveData();

            break;
              case K_F9:
gettext(22,4,37,4,NNN);

            for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

            No_Profile = atoi(Temp);
```

```
                gettext(22,5,37,5,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                No_Raysum = atoi(Temp);

                gettext(22,6,37,6,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                No_Section = atoi(Temp);

                gettext(60,4,75,4,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                AngularStep = atof(Temp);

                gettext(60,5,75,5,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                Ra_Interval = atof(Temp);

                gettext(60,6,75,6,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                Sec_Interval = atof(Temp);
//-

                if(indxS < (No_Profile - 2))
                                        {

for(i=0;i<12;i++) {

                                    if (NProfile[0][i]
!= ' ') Temp[i] = NProfile[0][i];

                                    else {Temp[i] =
'\0';break;}

                                    }

for(i=0;i<No_Profile;i++)

                sprintf(NProfile[i],"%s%i.dat\0",Temp
,i);
                                        }

                if(indxT < (No_Section - 2))
                                        {

for(i=0;i<12;i++) {

                                    if (NSection[0][i]
!= ' ') Temp[i] = NSection[0][i];

                                        else {Temp[i] =
'\0';break;}

                                        }

for(i=0;i<=No_Section+1;i++)

                sprintf(NSection[i],"%s%i.CTI",Temp
,i);
                                                }

//-

                quit=0;

                SaveData();

                edit(NConfig);
//
                GrInput();

                SaveData();

                break;
                    case K_ESC: quit =0; break;
                    case 0: break;
                    case BACKSPACE:
putch(even);putchar(' ');putch(even);break;
                    default: putch(even);break;
                     }
}while(quit);
}


void InputData(void)
{
int indxS=-1;
int indxT=-1;
int i;
int quit=1;
char NNN[4096];
char Temp[25];
time_t st,fn,cu;
unsigned even;
clrscr();
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqq\n");
for(i=0;i<23;i++)
printf("x
x\n");
printf("mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqq");
gotoxy(34,2);printf("INPUT CT DATA");
gotoxy(2,3);printf("-------------------------------
-------------------------");
gotoxy(2,4) ;printf("NO. of Projections :           ; ");
gotoxy(40,4);printf("Angular Step      :           ; ");
gotoxy(2,5) ;printf("NO. of ray-sum    :           ; ");
gotoxy(40,5);printf("Ray-sum interval  :           ; ");
gotoxy(2,6) ;printf("No. of Sections   :           ; ");
gotoxy(40,6);printf("Section interval  :           ; ");
gotoxy(2,7) ;printf("————————————————
————————————————");
gotoxy(2,8); printf("PROJECTIONS FILE
NAME(SOURCE FILE)");
gotoxy(51,8);printf("IMAGE FILE NAME(TARGET
FILE)");
gotoxy(2,9);printf("————————————————
————————————————");
gotoxy(40,9);printf("ß");
for(i=10;i<23;i++){gotoxy(40,i); printf("ßi");};
gotoxy(2,23);printf("————————————————
————————————————");
gotoxy(40,23); printf("ßi");
gotoxy(2,24);printf("<ESC> return MAINMENU ");
gotoxy(35,24);printf("<F-9> to edit");
gotoxy(60,24);printf("<F-10> to continue");
gotoxy(23,4);
do{
  cu = getEVEN(&even);
  switch(even){

                    case K_LEFT:
if((CurPosition.x>40)&&(CurPosition.y<15))
gotoxy(23,CurPosition.y);

if((CurPosition.x<40) && (CurPosition.y>4)
&&(CurPosition.y<15))

gotoxy(61,CurPosition.y-1);

if((CurPosition.x>40)&&(CurPosition.y>15))

gotoxy(5,20);

                break;
                    case K_RIGHT:
if(CurPosition.x<40)
                gotoxy(61,CurPosition.y);

if((CurPosition.x>40) &&(CurPosition.y <6) )

gotoxy(23,CurPosition.y+1);

if((CurPosition.x<40)&&(CurPosition.y>15))

gotoxy(45,20);

                break;
                    case K_UP: if((CurPosition.y >
4)&&(CurPosition.y<15))

gotoxy(CurPosition.x,CurPosition.y-1);

                if((CurPosition.y > 15) &&
(CurPosition.x>40))

gotoxy(61,6);

                if((CurPosition.y>15) &&
(CurPosition.x<40))

gotoxy(23,6);
                                break;
```

```
                    case K_DOWN: if((CurPosition.y
<6))

gotoxy(CurPosition.x,CurPosition.y+1);

if(CurPosition.y >=6) gotoxy(5,20);
                                break;
                    case K_ENTER:

                if((CurPosition.x<40)&&(CurPosition.
y>15))
                                      {
indxS++;gettext(5,20,29,20,NNN);

gotoxy(2,20);printf("%2i",indxS+1);

movetext(2,11,39,21,2,10);  gotoxy(5,20);

for(i=0;i<12;i++)

{NProfile[indxS][i] = NNN[i*2];
                                       }

NProfile[indxS][13] = '\0';
                                      }

                if((CurPosition.x>40)&&(CurPosition.
y>15))
                                      {
indxT++;gettext(45,20,69,20,NNN);

gotoxy(42,20);printf("%2i",indxT+1);

movetext(42,11,78,21,42,10); gotoxy(45,20);

for(i=0;i<12;i++)

{NSection[indxT][i] = NNN[i*2];
                                       }

NSection[indxT][13] ='\0';
                                      }

                break;
                    case
K_F10:gettext(22,4,37,4,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                No_Profile = atoi(Temp);

                gettext(22,5,37,5,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                No_Raysum = atoi(Temp);

                gettext(22,6,37,6,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';
```

```
                No_Section = atoi(Temp);

                gettext(60,4,75,4,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                AngularStep = atof(Temp);

                gettext(60,5,75,5,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                Ra_Interval = atof(Temp);

                gettext(60,6,75,6,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                Sec_Interval = atof(Temp);
//-

                if(indxS < (No_Profile - 2))
                                {

for(i=0;i<12;i++) {

                        if (NProfile[0][i]
!= ' ') Temp[i] = NProfile[0][i];

                        else {Temp[i] =
'\0';break;}

                        }

for(i=0;i<No_Profile;i++)

                sprintf(NProfile[i],"%s%i.pcx\0",Tem
p,i);

                                }

                if(indxT < (No_Section - 2))
                                {

for(i=0;i<12;i++) {

                        if (NSection[0][i]
!= ' ') Temp[i] = NSection[0][i];

                        else {Temp[i] =
'\0';break;}

                        }

for(i=0;i<=No_Section+1;i++)

                sprintf(NSection[i],"%s%i.CTI",Temp
,i);
                                }

//-
```

```
                quit=0;

                GrInput();

                SaveData();

                break;
                   case K_F9:
gettext(22,4,37,4,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                No_Profile = atoi(Temp);

                gettext(22,5,37,5,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                No_Raysum = atoi(Temp);

                gettext(22,6,37,6,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                No_Section = atoi(Temp);

                gettext(60,4,75,4,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                AngularStep = atof(Temp);

                gettext(60,5,75,5,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                Ra_Interval = atof(Temp);

                gettext(60,6,75,6,NNN);

                for(i=0;i<15;i++) Temp[i] =
NNN[i*2]; Temp[15] = '\0';

                Sec_Interval = atof(Temp);
//-

                if(indxS < (No_Profile - 2))
                                        {

for(i=0;i<12;i++) {

                                if (NProfile[0][i]
!= ' ') Temp[i] = NProfile[0][i];

                                else {Temp[i] =
'\0';break;}

                                }
for(i=0;i<No_Profile;i++)

                    sprintf(NProfile[i],"%s%i.pcx\0",Tem
p,i);
                                        }

                    if(indxT < (No_Section - 2))
                                        {

for(i=0;i<12;i++) {

                              if (NSection[0][i]
!= ' ') Temp[i] = NSection[0][i];

                              else {Temp[i] =
'\0';break;}

                              }

for(i=0;i<=No_Section+1;i++)

                    sprintf(NSection[i],"%s%i.CTI",Temp
,i);
                                        }

//-

                    quit=0;

                    SaveData();

                    edit(NConfig);

                    GrInput();

                    SaveData();

                    break;
                      case K_ESC: quit =0; break;
                      case 0: break;
                      case BACKSPACE:
putch(even);putchar(' ');putch(even);break;
                      default: putch(even);break;
                      }
}while(quit);
}

void IMain(int ret)
{
  switch(ret){
                         case 1:

                    gotoxy(30,13);textattr(135);cprintf("1.
INPUT CT DATA");

                    gotoxy(30,14);textattr(007);cprintf("2.
PROFILES DATA FROM FILES");

                    gotoxy(30,15);textattr(007);cprintf("3.
RECONSTRUCTION OF CT IMAGE");

                    gotoxy(30,16);textattr(007);cprintf("4.
DISPLAY THE CT IMAGE");

                    gotoxy(30,17);textattr(007);cprintf("5.
EXIT THIS SOFTWARE");
                                        break;
                         case 2:

                    gotoxy(30,13);textattr(007);cprintf("1.
INPUT CT DATA");

                    gotoxy(30,14);textattr(135);cprintf("2.
PROFILES DATA FROM FILES");

                    gotoxy(30,15);textattr(007);cprintf("3.
RECONSTRUCTION OF CT IMAGE");

                    gotoxy(30,16);textattr(007);cprintf("4.
DISPLAY THE CT IMAGE");

                    gotoxy(30,17);textattr(007);cprintf("5.
EXIT THIS SOFTWARE");
                                        break;
                         case 3:

                    gotoxy(30,13);textattr(007);cprintf("1.
INPUT CT DATA");

                    gotoxy(30,14);textattr(007);cprintf("2.
PROFILES DATA FROM FILES");

                    gotoxy(30,15);textattr(135);cprintf("3.
RECONSTRUCTION OF CT IMAGE");

                    gotoxy(30,16);textattr(007);cprintf("4.
DISPLAY THE CT IMAGE");

                    gotoxy(30,17);textattr(007);cprintf("5.
EXIT THIS SOFTWARE");
                                        break;
                         case 4:

                    gotoxy(30,13);textattr(007);cprintf("1.
INPUT CT DATA");

                    gotoxy(30,14);textattr(007);cprintf("2.
PROFILES DATA FROM FILES");

                    gotoxy(30,15);textattr(007);cprintf("3.
RECONSTRUCTION OF CT IMAGE");

                    gotoxy(30,16);textattr(135);cprintf("4.
DISPLAY THE CT IMAGE");

                    gotoxy(30,17);textattr(007);cprintf("5.
EXIT THIS SOFTWARE");
                                        break;
                         case 5:

                    gotoxy(30,13);textattr(007);cprintf("1.
INPUT CT DATA");

                    gotoxy(30,14);textattr(007);cprintf("2.
PROFILES DATA FROM FILES");
```

```
                gotoxy(30,15);textattr(007);cprintf("3.
RECONSTRUCTION OF CT IMAGE");

                gotoxy(30,16);textattr(007);cprintf("4.
DISPLAY THE CT IMAGE");

                gotoxy(30,17);textattr(135);cprintf("5.
EXIT THIS SOFTWARE");
                                break;
                        default: break;
                        }
gotoxy(44,20);
}


int MainMenu(int cc)
{
int i,quit;
int ret;
unsigned key;
clrscr();

printf(";qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqq\n");
for(i=0;i<23;i++)
printf("3
3\n");
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqq");
gotoxy(2,2);printf("Thesis Title : DEVELOPMENT OF
THREE DIMENSIONAL COMPUTED
TOMOGRAPHY SOFTWARE");
gotoxy(2,3);printf("By              : MR. CHERNOCHY
SOYPETCH");
gotoxy(2,4);cprintf("Department       : NUCLEAR
TECHNOLOGY");
gotoxy(2,5);printf("Thesis Advisor    : ASST. PROF.
SOMYOT SRISATIT");
gotoxy(2,6);cprintf("Thesis Co-advisor : -");
gotoxy(2,7);cprintf("-------------------------------------
----------------------------");
gotoxy(36,10);cprintf("MAIN MENU");
if (!cc) {gotoxy(30,14);cprintf("2. PROFILES DATA
FROM FILES");}
if (!cc) {gotoxy(30,15);cprintf("3.
RECONSTRUCTION OF CT IMAGE");}
if (!cc) {gotoxy(30,16);cprintf("4. DISPLAY THE CT
IMAGE");}
if (!cc) {gotoxy(30,17);cprintf("5. EXIT THIS
SOFTWARE");}
gotoxy(2,22);cprintf("-------------------------------------
----------------------------");
gotoxy(10,23);cprintf("Copyright 1997 by Graduate
School Chulalongkorn University.");
gotoxy(25,24);cprintf("All rights reserved worldwide.");
if (!cc) {gotoxy(30,13);textattr(135);cprintf("1. INPUT
CT DATA");}
gotoxy(34,20);cprintf("Select :- ");
ret = cc;quit=1;
  clr_KB_BUFFER();
if (cc != 0) IMain(cc);
```

```
key = 0;
do{
  get_KB_BUFFER(&key);
  switch(key){
                        case K_UP:
                                                        ret =
ret-1;

                        if(ret < 1) ret = 5;

                        IMain(ret);

                        break;
                        case K_DOWN:
                                                        ret =
ret+1;

                        if(ret>5) ret = 1;

                        IMain(ret);

                        break;
                        case 0x4f31:
                        case 0x231:
                                        ret = 1; quit =
0;printf("%i",ret);
                                        break;
                        case 0x5032:
                        case 0x332:
                                        ret = 2; quit =
0;printf("%i",ret);
                                        break;
                        case 0x433:
                        case 0x5133:
                                        ret = 3; quit =
0;printf("%i",ret);
                                        break;
                        case 0x534:
                        case 0x4b34:
                                        ret = 4; quit =
0;printf("%i",ret);
                                        break;
                        case 0x4c35:
                        case 0x635:
                                        ret = 5; quit =
0;printf("%i",ret);
                                        break;
                        case K_TAB: IDENPASS =
!IDENPASS; break;
                        case K_ENTER: quit = 0;
printf("%i",ret);
                        default: break;
                        }
  key=0;
}while(quit==1);
textattr(7);
return ret;
}

void SMain(int ret)
{
  switch(ret){
                        case 1:
```

```
                gotoxy(30,13);textattr(135);cprintf("1.
SHEPP-LOGAN");

                gotoxy(30,14);textattr(007);cprintf("2.
RAM-LAK");

                gotoxy(30,15);textattr(007);cprintf("3.
RETURN TO MAIN MENU");
                                break;
                case 2:

                gotoxy(30,13);textattr(007);cprintf("1.
SHEPP-LOGAN");

                gotoxy(30,14);textattr(135);cprintf("2.
RAM-LAK");

                gotoxy(30,15);textattr(007);cprintf("3.
RETURN TO MAIN MENU");
                                break;
                case 3:

                gotoxy(30,13);textattr(007);cprintf("1.
SHEPP-LOGAN");

                gotoxy(30,14);textattr(007);cprintf("2.
RAM-LAK");

                gotoxy(30,15);textattr(135);cprintf("3.
RETURN TO MAIN MENU");
                                break;
                                default: break;
            }
gotoxy(44,20);
}


int FilterMenu(int oo)
{
int i,quit;
int ret;
unsigned key;

clrscr();
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqw\n");
for(i=0;i<23;i++)
printf("ß
ß\n");
printf("ŋqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqq");
gotoxy(2,2);printf("Thesis Title : DEVELOPMENT OF
THREE DIMENSIONAL COMPUTED
TOMOGRAPHY SOFTWARE");
gotoxy(2,3);printf("By              : MR. CHERNGCHY
SOYPETCH");
gotoxy(2,4);printf("Department       : NUCLEAR
TECHNOLOGY");
gotoxy(2,5);printf("Thesis Advisor   : ASST. PROF.
SOMYOT SRIBATIT");
gotoxy(2,6);printf("Thesis Co-advisor : -");
gotoxy(2,7);printf("------------------------------------
--------------------------");
gotoxy(28,10);cprintf("SELECT FILTER
FUNCTION");
if(!oo) {gotoxy(30,14);cprintf("2. RAM-LAK");}
if(!oo) {gotoxy(30,15);cprintf("3. RETURN TO MAIN
MENU");}
gotoxy(2,22);printf("------------------------------------
--------------------------");
gotoxy(10,23);printf("Copyright 1997 by Graduate
School Chulalongkorn University.");
gotoxy(25,24);printf("All rights reserved worldwide.");
if(!oo) {gotoxy(30,13);textattr(135);cprintf("1. SHEPP-
LOGAN");}
gotoxy(34,20);printf("Select :- ");
ret = oo;quit=1;
  SMain(ret);
  clr_KB_BUFFER();
  key = 0;
do{
   get_KB_BUFFER(&key);
   switch(key){
                        case K_UP:
                                                        ret =
ret-1;

                    if(ret < 1) ret = 3;

                    SMain(ret);

                    break;
                        case K_DOWN:
                                                        ret =
ret+1;

                    if(ret>3) ret = 1;

                    SMain(ret);

                    break;
                        case 0x4f31:
                        case 0x231:
                                        ret = 1; quit =
0;printf("%i",ret);
                                        break;
                        case 0x5032:
                        case 0x332:
                                        ret = 2; quit =
0;printf("%i",ret);
                                        break;
                        case 0x433:
                        case 0x5133:
                                        ret = 3; quit =
0;printf("%i",ret);
                                        break;
                        case 0x534:
                        case 0x4b34:
                                        ret = 4; quit =
0;printf("%i",ret);
                                        break;
                        case K_ENTER: quit = 0;
printf("%i",ret);
                        default: break;
                        }
```

```
 key=0;
}while(quit==1);
textattr(7);
return ret;

}



void DMain(int ret)
{
  switch(ret){
                    case 1:

                gotoxy(30,13);textattr(135);cprintf("1.
DISPLAY ALL");

                gotoxy(30,14);textattr(007);cprintf("2.
DISPLAY PROFILE");

                gotoxy(30,15);textattr(007);cprintf("3.
DISPLAY 2 DIMENSIONAL SECTION");

                gotoxy(30,16);textattr(007);cprintf("4.
DISPLAY 3 DIMENSIONAL PICTURE");

                gotoxy(30,17);textattr(007);cprintf("5.
RETURN MAIN MENU");
                                  break;
                    case 2:

                gotoxy(30,13);textattr(007);cprintf("1.
DISPLAY ALL");

                gotoxy(30,14);textattr(135);cprintf("2.
DISPLAY PROFILE");

                gotoxy(30,15);textattr(007);cprintf("3.
DISPLAY 2 DIMENSIONAL SECTION");

                gotoxy(30,16);textattr(007);cprintf("4.
DISPLAY 3 DIMENSIONAL PICTURE");

                gotoxy(30,17);textattr(007);cprintf("5.
RETURN MAIN MENU");
                                  break;
                    case 3:

                gotoxy(30,13);textattr(007);cprintf("1.
DISPLAY ALL");

                gotoxy(30,14);textattr(007);cprintf("2.
DISPLAY PROFILE");

                gotoxy(30,15);textattr(135);cprintf("3.
DISPLAY 2 DIMENSIONAL SECTION");

                gotoxy(30,16);textattr(007);cprintf("4.
DISPLAY 3 DIMENSIONAL PICTURE");

                gotoxy(30,17);textattr(007);cprintf("5.
RETURN MAIN MENU");
                                  break;
                    case 4:

                gotoxy(30,13);textattr(007);cprintf("1.
DISPLAY ALL");

                gotoxy(30,14);textattr(007);cprintf("2.
DISPLAY PROFILE");

                gotoxy(30,15);textattr(007);cprintf("3.
DISPLAY 2 DIMENSIONAL SECTION");

                gotoxy(30,16);textattr(135);cprintf("4.
DISPLAY 3 DIMENSIONAL PICTURE");

                gotoxy(30,17);textattr(007);cprintf("5.
RETURN MAIN MENU");
                                  break;
                    case 5:

                gotoxy(30,13);textattr(007);cprintf("1.
DISPLAY ALL");

                gotoxy(30,14);textattr(007);cprintf("2.
DISPLAY PROFILE");

                gotoxy(30,15);textattr(007);cprintf("3.
DISPLAY 2 DIMENSIONAL SECTION");

                gotoxy(30,16);textattr(007);cprintf("4.
DISPLAY 3 DIMENSIONAL PICTURE");

                gotoxy(30,17);textattr(135);cprintf("5.
RETURN MAIN MENU");
                                  break;

                                  default: break;
           }
gotoxy(44,20);
}



int DispMenu(int co)
{
int i,quit;
int ret;
unsigned key;

clrscr();
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqw\n");
for(i=0;i<23;i++)
printf("x
x\n");
printf("mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqq");
gotoxy(2,2);printf("Thesis Title : DEVELOPMENT OF
THREE DIMENSIONAL COMPUTED
TOMOGRAPHY SOFTWARE");
gotoxy(2,3);printf("By             : MR. CHERNGCHY
SOYPETCH");
gotoxy(2,4);printf("Department     : NUCLEAR
TECHNOLOGY");
```

```
gotoxy(2,5);printf("Thesis Advisor   : ABST. PROF.
SOMYOT SRISATIT");
gotoxy(2,6);printf("Thesis Co-advisor : -");
gotoxy(2,7);printf("——————————————————
————————————————");
gotoxy(34,10);printf("DISPLAY MENU");
if(!cc) {gotoxy(30,14);cprintf("2. DISPLAY
PROFILE");}
if(!cc) {gotoxy(30,15);cprintf("3. DISPLAY 2
DIMENSIONAL SECTION");}
if(!cc) {gotoxy(30,16);cprintf("4. DISPLAY 3
DIMENSIONAL PICTURE");}
if(!cc) {gotoxy(30,17);cprintf("5. RETURN MAIN
MENU");}
gotoxy(2,22);printf("——————————————————
————————————————");
gotoxy(10,23);printf("Copyright 1997 by Graduate
School Chulalongkorn University.");
gotoxy(25,24);printf("All rights reserved worldwide.");
if(!cc) {gotoxy(30,13);textattr(135);cprintf("1.
DISPLAY ALL");}
gotoxy(34,20);cprintf("Select :- ");
ret = cc;quit=1;
 DMain(ret);
 clr_KB_BUFFER();
 key = 0;
do{
 get_KB_BUFFER(&key);
 switch(key){
              case K_UP:
                                   ret =
ret-1;

           if(ret < 1) ret = 5;

           DMain(ret);

           break;
              case K_DOWN:
                                   ret =
ret+1;

           if(ret>5) ret = 1;

           DMain(ret);

           break;
              case 0x4f31:
              case 0x231:
                              ret = 1; quit =
0;printf("%i",ret);
                              break;
              case 0x5032:
              case 0x332:
                              ret = 2; quit =
0;printf("%i",ret);
                              break;
              case 0x433:
              case 0x5133:
                              ret = 3; quit =
0;printf("%i",ret);
                              break;
              case 0x534:
              case 0x4b34:
                                   ret = 4; quit =
 0;printf("%i",ret);
                                   break;
                    case 0x4c35:
                    case 0x635:
                                   ret = 5; quit =
 0;printf("%i",ret);
                                   break;
                    case K_ENTER: quit = 0;
 printf("%i",ret);
                    default: break;
                   }
  key=0;
 }while(quit==1);
 textattr(7);
 return ret;

}



void Identifier(void)
{
FILE *in, *in2;
char temp[255];
int i,j;
struct palettetype pal;
fpos_t *pos;
float v,k;
int f,c;
unsigned even,q=1;

clrscr();
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqq\n");
for(j=0;j<23;j++)
printf("x
x\n");
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqq");
gotoxy(2,2);printf("Thesis Title : DEVELOPMENT OF
THREE DIMENSIONAL COMPUTED
TOMOGRAPHY SOFTWARE");
gotoxy(2,3);printf("By              : MR. CHERNGCHY
SOYPETCH");
gotoxy(2,4);printf("Department      : NUCLEAR
TECHNOLOGY");
gotoxy(2,5);printf("Thesis Advisor   : ABST. PROF.
SOMYOT SRISATIT");
gotoxy(2,6);printf("Thesis Co-advisor : -");
gotoxy(2,7);printf("——————————————————
————————————————");
gotoxy(2,22);printf("——————————————————
————————————————");
gotoxy(10,23);printf("Copyright 1997 by Graduate
School Chulalongkorn University.");
gotoxy(25,24);printf("All rights reserved worldwide.");
gotoxy(23,10);printf("INPUT IDENTIFIER FILE
NAME SCREEN");

NEWIN:;
```

```
gotoxy(24,14);printf("INPUT FILE NAME -: ");
scanf("%s",NConfig);
if((strcmp(NConfig,"q") == 0) || (strcmp(NConfig,"Q")
==0)) goto PASS;
                    if ((in = fopen(NConfig, "rt"))   ==
NULL)
                       { gotoxy(23,17);printf("Cannot open
input file.");
                        goto NEWIN;
                       }
   IDENPASS =1;
   for (i = 0; i<250; i++){
                                          temp[i] = fgetc(in);
                                          if (temp[i] == '\n' ) {j
= i;i = 250;}
                                                               }
                                          temp[j] = '\0';
   for (i = 0; i < strlen(temp); i++ ) {
                                          switch (temp[i]) {
                                                              case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                      }

                                         }

  lclrblank(temp);
  No_Profile = atoi(temp);
   for (i = 0; i<250; i++){
                                          temp[i] = fgetc(in);
                                          if (temp[i] == '\n' ) {j
= i;i = 250;}
                                                               }
                                          temp[j] = '\0';
   for (i = 0; i < strlen(temp); i++ ) {
                                          switch (temp[i]) {
                                                              case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                      }

                                         }

  lclrblank(temp);
  No_Raysum = atoi(temp);
   for (i = 0; i<250; i++){
                                          temp[i] = fgetc(in);
                                          if (temp[i] == '\n' ) {j
= i;i = 250;}
                                                               }
                                          temp[j] = '\0';
   for (i = 0; i < strlen(temp); i++ ) {
                                          switch (temp[i]) {
                                                              case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                      }
                                         }

  lclrblank(temp);
  No_Section = atoi(temp);
   for (i = 0; i<250; i++){
                                          temp[i] = fgetc(in);
                                          if (temp[i] == '\n' ) {j
= i;i = 250;}
                                                               }
                                          temp[j] = '\0';
   for (i = 0; i < strlen(temp); i++ ) {
                                          switch (temp[i]) {
                                                              case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                      }

                                         }

  lclrblank(temp);
  AngularStep = atoi(temp);
   for (i = 0; i<250; i++){
                                          temp[i] = fgetc(in);
                                          if (temp[i] == '\n' ) {j
= i;i = 250;}
                                                               }
                                          temp[j] = '\0';
   for (i = 0; i < strlen(temp); i++ ) {
                                          switch (temp[i]) {
                                                              case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                      }

                                         }

  lclrblank(temp);
  Ra_Interval = atoi(temp);
   for (i = 0; i<250; i++){
                                          temp[i] = fgetc(in);
                                          if (temp[i] == '\n' ) {j
= i;i = 250;}
                                                               }
                                          temp[j] = '\0';
   for (i = 0; i < strlen(temp); i++ ) {
                                          switch (temp[i]) {
                                                              case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                      }

                                         }

  lclrblank(temp);
  Sec_Interval = atoi(temp);
   for (i = 0; i<250; i++){
                                          temp[i] = fgetc(in);
```

```
                              if (temp[i] == '\n' ) {j
= i;i = 250;}
                                    }
                              temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                              switch (temp[i]) {
                                    case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                    }

                              }
 lstrblank(temp);
 PicMaxX = atoi(temp);
  for (i = 0; i<250; i++){
                              temp[i] = fgetc(in);
                              if (temp[i] == '\n' ) {j
= i;i = 250;}
                                    }
                              temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                              switch (temp[i]) {
                                    case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                    }

                              }
 lstrblank(temp);
 PicMaxY = atoi(temp);
  for (i = 0; i<250; i++){
                              temp[i] = fgetc(in);
                              if (temp[i] == '\n' ) {j
= i;i = 250;}
                                    }
                              temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                              switch (temp[i]) {
                                    case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                    }

                              }
 lstrblank(temp);
 PicLeft = atoi(temp);
  for (i = 0; i<250; i++){
                              temp[i] = fgetc(in);
                              if (temp[i] == '\n' ) {j
= i;i = 250;}
                                    }
                              temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                              switch (temp[i]) {
```

```
                                                            case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                    }

                              }
 lstrblank(temp);
 PicTop = atoi(temp);
  for (i = 0; i<250; i++){
                              temp[i] = fgetc(in);
                              if (temp[i] == '\n' ) {j
= i;i = 250;}
                                    }
                              temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                              switch (temp[i]) {
                                    case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                    }

                              }
 lstrblank(temp);
 PicRight = atoi(temp);
  for (i = 0; i<250; i++){
                              temp[i] = fgetc(in);
                              if (temp[i] == '\n' ) {j
= i;i = 250;}
                                    }
                              temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                              switch (temp[i]) {
                                    case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                    }

                              }
 lstrblank(temp);
 PicBottom = atoi(temp);
  for (i = 0; i<250; i++){
                              temp[i] = fgetc(in);
                              if (temp[i] == '\n' ) {j
= i;i = 250;}
                                    }
                              temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                              switch (temp[i]) {
                                    case
'=': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                    }

                              }
  lclrblank(temp);
  minAll = atof(temp);
   for (i = 0; i<250; i++){
                              temp[i] = fgetc(in);
                              if (temp[i] == '\n' ) {j
= i;i = 250;}
                                   }
                              temp[j] = '\0';
   for (i = 0; i < strlen(temp); i++ ) {
                              switch (temp[i]) {
                                                  case
'\t': temp[i] = ' ';i = strlen(temp); break;

                    default: temp[i] = ' ';break;

                    }

                              }
  lclrblank(temp);
  maxAll = atof(temp);

  fscanf(in,"%s",temp);
  fscanf(in,"%s",temp);
  for(i=0;i<No_Profile;i++) fscanf(in,"%s",NProfile[i]);
  fscanf(in,"%s",temp);
  fscanf(in,"%s",temp);
  for(i=0;i<No_Section;i++)
fscanf(in,"%s",NSection[i]);
                    fclose(in);

//
                    i = No_Section/2;
                    strcpy(targetFile,NSection[i]);
                    if ((in = fopen(targetFile, "rt")) ==
NULL)
                    {
                      fprintf(stderr, "Cannot open input \
file.\n");
                      exit(1);
                    }
                    Readdata();
                    VGA256graph();
                    BW192();
                    setcolor(15);
                    k = 300/raNO;
                    line(400,10,592,10);
line(400,26,592,26);
                    line(400,10,400,26);
line(592,10,592,26);
                    for(i=0;i<192;i++){
                              setcolor(i+64);

                    line(401+i,11,401+i,25);
                    }
                    setcolor(15);
                    sprintf(temp,"Max : %5.2f",maxAll);
                    outtextxy(330,1,temp);
                    sprintf(temp,"Min : %5.2f",minAll);
                    outtextxy(545,29,temp);
//                  sprintf(temp,"<F1> : NO, Calculate
Air");
//                  outtextxy(400,100,temp);
//                  sprintf(temp,"<F2> : NO, Fill
Section");
```

```
//                  outtextxy(400,120,temp);
                    sprintf(temp,"<F3> : Positive
Picture");
                    outtextxy(400,140,temp);
                    outtextxy(230,465,"Press Enter to
continue");
                    Openwindow(10,10,310,310);
                    fseek(in,0,SEEK_SET);
                    j=0;
                    for(i=0;i<400;i++)
                    {
                              temp[i] = fgetc(in);
                              if (temp[i] == '\n')
j++;
                              if (j == 5) i = 400;
                    }

                    for(i=0;i<raNO;i++)
                                        {

                    for (j=0;j<raNO;j++)

                    {

                              fscanf(in,"%f",&v);

                              if(v != -1000)

                              {

                                        f=
((maxAll-v) * (189 / (maxAll-minAll))) + 1;


                    putpixel(i*k+1,j*k+1,f+64);

                              }

                    }
                                        }
                    fclose(in);
                    setcolor(15);
                    setrgbpalette(18,0,0,0);
                    setrgbpalette(19,60,60,60);
                    setviewport(0,0,639,479,1);
do{
  getEVEN(&even);
  switch(even){
                              case 0: break;
                              case K_F1:

                    AirStatus = !AirStatus;

switch(AirStatus)

{

                    case 0:

                      setcolor(18);

                      sprintf(temp,"<F1> : Calculate
Air0");
```

```
                outtextxy(400,100,temp);
                setcolor(19);
                sprintf(temp,"<F1> : NO,
Calculate Air\0");
                outtextxy(400,100,temp);
                break;
            case 1:
                setcolor(18);
                sprintf(temp,"<F1> : NO,
Calculate Air\0");
                outtextxy(400,100,temp);
                setcolor(19);
                sprintf(temp,"<F1> : Calculate
Air\0");
                outtextxy(400,100,temp);
                break;
            default:   break;
}
break;
                        case K_F2: FillS =
!FillS;
switch(FillS)
                                {
            case 0:
                setcolor(18);
                        sprintf(temp,"<F2> :
Fill Section");
            outtextxy(400,120,temp);
                setcolor(19);
                        sprintf(temp,"<F2> :
NO, Fill Section");
            outtextxy(400,120,temp);
                break;
            case 1:
                setcolor(18);
```

```
                                        sprintf(temp,"<F2> :
NO, Fill Section");
                            outtextxy(400,120,temp);
                                setcolor(19);
                                        sprintf(temp,"<F2> :
Fill Section");
                            outtextxy(400,120,temp);
                            default :break;
                                                        }
break;
                                        case K_F3: RealOb
= !RealOb;
switch(RealOb)
                                                        {
case 0:
                                setcolor(18);
                                        sprintf(temp,"<F3> :
Positive Picture");
                            outtextxy(400,140,temp);
                                setcolor(19);
                                        sprintf(temp,"<F3> :
Negative Picture");
                            outtextxy(400,140,temp);
                                break;
case 1:
                                setcolor(18);
                                        sprintf(temp,"<F3> :
Negative Picture");
                            outtextxy(400,140,temp);
                                setcolor(19);
                                        sprintf(temp,"<F3> :
Positive Picture");
                            outtextxy(400,140,temp);
                                break;
```

```
                    default: break;
                                          }
break;
                                    case K_ENTER: q =
!q; break;
                                    default: break;
                                   }
}while(q);
          -         closegraph();
PASS:;

}




char far cc[60][256];

time_t  t_start,t_finish;



void main(void)
{
int gdriver;
int gmode;
int errorcode;
int i,ii;
int s,d,dd;
unsigned even;

char tmp[20];
for(i=0;i<60;i++)
{
 sprintf(tmp,"A%i.CTI",i);
 strcpy(NSection[i],tmp);
 sprintf(tmp,"A%i.PCX",i);
 strcpy(NProfile[i],tmp);
}
//No_Section = 20;
//                    PicMaxX = 255; PicMaxY = 255;
//                    PicLeft = 0; PicRight = PicMaxX;
//                    PicTop = 0; PicBottom = PicMaxY;
i =1; s =1; d = 1;
do{
 i = MainMenu(i);
   switch(i){
                      case 1: FormFile = 0;
                                    InputData();i=1;
                                    rsNO =
No_Raysum;
                                    pjNO =
No_Profile;

rayInterval=Ra_Interval,

angularStep=AngularStep;
                                    break;
                      case 2: FormFile = 1;
                                    InputPro();
                                    rsNO =
No_Raysum;
```

```
                                    pjNO =
No_Profile;

rayInterval=Ra_Interval,

angularStep=AngularStep;
                                    break;
                      case 3:
//                                  corn_tur();
                                    do{
                                     s =
FilterMenu(s);
                                     switch(s){

                 case 1:

                 case 2: corn_tur(s);

                                  printf("\r\npress any
key to continue");

                                  getch();

                                  break;

                 default:

                     break;

                 }
                                    }while(s!=3);
                                    break;
                      case 4:
//                                  gr3d();
                                    do{
                                    d = DispMenu(d);
                                    Ax =0; Ay =0; Az
=0;

                                    switch(d){
//
case 1:  Identifier();

case 1:  showall();

                 break;

case 2:  showprofile();

                 break;

case 3:

                 gr3dI();

                 Dis2Pic();
//
                 drawVGA64();

                 break;

case 4: gr3d();

                 break;
case 5:

default: break;
                    }
            }while(d!=5);
            break;
```

```
                    default: break;
                    }
            }while(i!=5);

            clrscr();

            }
```

SOURCE CODE OF CT.H

```
/*PCX Read*/
extern struct PCXHEAD {
                    char manufacturer;
                    char version;
                    char encoding;
                    char bit_per_pixel;
                    int xmin,ymin;
                    int xmax,ymax;
                    int hres;
                    int vres;
                    char palette[48];
                    char reserved;
                    char colour_planes;
                    int bytes_per_line;
                    int palette_type;
                    char filler[58];
               };
float far fp[15360];
//float   *fp;
float   far ct[60][256];

char sourceFile[250], targetFile[250];
char ctFile[250];
int  pjNO,raNO;  /* projection NO and ray-sum NO */
int  Mode_vdo;
float rayInterval, angularStep;


float max_value, min_value;



void lclrblank(char *str)
{
char c[250];
int i,j;
          for (i=0;i<strlen(str);i++)
          {
                    if (str[i] != ' ') {
                                        strcpy(c,str + i);
                                        i = strlen(str);
                                         }
          }
          strcpy(str,c);
}

void readINI(void)
{
FILE *in;
char temp[250],c;
int i,j;
//strcpy(ctFile,NConfig);
if ((in = fopen(ctFile, "rt"))== NULL)
{
  fprintf(stderr, "Cannot open input \file.\n");
  exit(1);
}
  for (i = 0; i<250; i++){
                    temp[i] = fgetc(in);
                    if (temp[i] == '\n' ) {j = i;i = 250;}
                                   }
                    temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                    switch (temp[i]) {
                              case '=': temp[i] = ' ';i = strlen(temp);
break;
                              default: temp[i] = ' ';break;
                                        }
                                                  }
  strcpy(sourceFile,temp);
  lclrblank(sourceFile);

  for (i = 0; i<250; i++){
                    temp[i] = fgetc(in);
                    if (temp[i] == '\n' ) {j = i;i = 250;}
                                   }
                    temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                    switch (temp[i]) {
                              case '=': temp[i] = ' ';i = strlen(temp);
break;
                              default: temp[i] = ' ';break;
                                        }
                                                  }
          strcpy(targetFile,temp);
          lclrblank(targetFile);

  for (i = 0; i<250; i++){
                    temp[i] = fgetc(in);
                    if (temp[i] == '\n' ) {j = i;i = 250;}
                                   }
                    temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                    switch (temp[i]) {
                              case '=': temp[i] = ' ';i = strlen(temp);
break;
                              default: temp[i] = ' ';break;
                                        }
                                                  }
  lclrblank(temp);
  pjNO = atoi(temp);

  for (i = 0; i<250; i++){
                    temp[i] = fgetc(in);
                    if (temp[i] == '\n' ) {j = i;i = 250;}
                                   }
                    temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                    switch (temp[i]) {
                              case '=': temp[i] = ' ';i = strlen(temp);
break;
                              default: temp[i] = ' ';break;
                                        }
                                                  }
  lclrblank(temp);
  raNO = atoi(temp);

  for (i = 0; i<250; i++){
                    temp[i] = fgetc(in);
                    if (temp[i] == '\n' ) {j = i;i = 250;}
                                   }
                    temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                    switch (temp[i]) {
                        case '=': temp[i] = ' ';i = strlen(temp);
break;
                        default: temp[i] = ' ';break;
                              }
                                   }
  lstrblank(temp);
  rayInterval = atof(temp);

  for (i = 0; i<250; i++){
                  temp[i] = fgetc(in);
                  if (temp[i] == '\n' ) {j = i;i = 250;}
                        }
                  temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                  switch (temp[i]) {
                        case '=': temp[i] = ' ';i = strlen(temp);
break;
                        default: temp[i] = ' ';break;
                              }
                                   }
  lstrblank(temp);
  angularStep = atof(temp);


  for (i = 0; i<250; i++){
                  temp[i] = fgetc(in);
                  if (temp[i] == '\n' ) {j = i;i = 250;}
                        }
                  temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                  switch (temp[i]) {
                        case '=': temp[i] = ' ';i = strlen(temp);
break;
                        default: temp[i] = ' ';break;
                              }
                                   }
  lstrblank(temp);
  Mode_vdo = atoi(temp);

  for (i = 0; i<250; i++){
                  temp[i] = fgetc(in);
                  if (temp[i] == '\n' ) {j = i;i = 250;}
                        }
                  temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                  switch (temp[i]) {
                        case '=': temp[i] = ' ';i = strlen(temp);
break;
                        default: temp[i] = ' ';break;
                              }
                                   }
  lstrblank(temp);
  max_value = atof(temp);

  for (i = 0; i<250; i++){
                  temp[i] = fgetc(in);
                  if (temp[i] == '\n' ) {j = i;i = 250;}
                        }
                  temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                  switch (temp[i]) {
                        case '=': temp[i] = ' ';i = strlen(temp);
break;
                        default: temp[i] = ' ';break;
```

```
                                        }
                                                  }
  lstrblank(temp);
  min_value = atof(temp);


fclose(in);
}

void Convolution(void)
{
unsigned long i,j,l;
unsigned long p;
float shepp,s,s1,sum;
float pi2,kr,kr2,sg;
FILE *out;

          pi2 = Pi * Pi;
          sum =0;
          shepp = 2.0 / (pi2 * rayInterval);
          for(i=0;i<pgNO;i++) {
                    p = i/pgNO * 100;
                    printf(" ");
           for(j=0;j<raNO;j++) {
                    sum = 0.0;
                    for(l=0;l<raNO;l++) {
                              s1 = 1.0 - 4.0 * ((j-l) * (j-l));
                      sg = shepp / s1;
                              sum += fp[l+raNO* i] * sg;
                    }
            ct[i][j] = sum;
           }
          }
          free(fp);

if ((out = fopen("origin.cal", "wt"))           == NULL)
{
  fprintf(stderr, "Cannot open output file \ file.\n");
  exit(1);
}
          for(i=0;i<pgNO;i++) {
                    for(j=0;j<raNO;j++) {

                    fprintf(out,"%5.4f\n",ct[i][j]);

                    }

          }
fclose(out);
}

void SheppLogan(void)
{
unsigned long i,j,l;
unsigned long p;
float shepp,s,s1,sum;
float pi2,kr,kr2,sg;
FILE *out;

          pi2 = Pi * Pi;
          sum =0;
          shepp = 2.0 / (pi2 * rayInterval);
          for(i=0;i<pgNO;i++) {
                    p = i/pgNO * 100;
                    printf(" ");
              for(j=0;j<raNO;j++) {
                    sum = 0.0;
                    for(l=0;l<raNO;l++) {
                              s1 = 1.0 - 4.0 * ((j-l) * (j-l));
                      sg = shepp / s1;
                              sum += fp[l+raNO* i] * sg;
                    }
               ct[i][j] = sum;
              }
             }
             free(fp);

if ((out = fopen("Shepp.cal", "wt"))                == NULL)
{
  fprintf(stderr, "Cannot open output file \ file.\n");
  exit(1);
}
             for(i=0;i<pjNO;i++) {
                    for(j=0;j<raNO;j++) {

                    fprintf(out,"%5.4f\n",ct[i][j]);

                    }

             }
fclose(out);
}

void RamLak(void)
{
unsigned long i,j,l;
unsigned long p;
float shepp,a,s1,sum;
float pi2,kr,kr2,sg;
FILE *out;

             pi2 = Pi * Pi;
             sum =0.0;
             shepp = 1.0 / (pi2 * rayInterval);
             for(i=0;i<pjNO;i++) {
                    p = i/pjNO * 100;
                    printf(" ");
              for(j=0;j<raNO;j++) {
                    sum = 0.0;
                    for(l=0;l<raNO;l++) {
                              if((j-l) == 0) sg = 0.25 / rayInterval;
                              if((j-l) != 0) {
                                                  s1 =
-1*sin(Pi*(j-l)/2)*sin(Pi*(j-l)/2)/((j-l)*(j-l));
                                                  sg =
shepp *s1;
                                           }
                              sum += fp[l+raNO* i] * sg;
                    }
               ct[i][j] = sum;
              }
             }
             free(fp);

if ((out = fopen("ramlak.cal", "wt"))               == NULL)
{
  fprintf(stderr, "Cannot open output file \ file.\n");
  exit(1);
}
             for(i=0;i<pjNO;i++) {
                    for(j=0;j<raNO;j++) {

                    fprintf(out,"%5.4f\n",ct[i][j]);

                    }


             }
fclose(out);
}

double Pa(void)
{
             if( pjNO * angularStep <= 180.0)
              return(Pi);
             else
              return(Pi * 2);
}

double midpoint(void)
{
                    return ( (raNO - 1) / 2 );
}

double mid2point(void)
{
             return (((raNO-1)/2) * ((raNO-1)/2) );
}

double angle(float l)
{
             return( radian *l * angularStep);
}

int incycle(int i,int j)
{
int ret;
double ix,jy,xy;
                              ix = i - ((raNO-1)/2);
                              jy = j - ((raNO-1)/2);
                              xy = ix*ix + jy*jy;
                              if(xy > mid2point()) ret = 0;
                              else ret = 1;
                              return ret;
}

double rpoint(int i,int j,double ang)
{
double ret;
double x,y;
                              x = j - ((raNO-1)/2);
                              y = i - ((raNO-1)/2);
                              ret = ((raNO-1)/2) + x*cos(ang) +
y*sin(ang);
             return ret;
}

void backprojection(void)
{
long i,j,k,p,q;
int percent;
```

```
float g_value,minx,maxx;
double r_point,dif_r,point_lower,point_upper;
float s_value;
float ang;
float pie;
FILE *out;

if ((out = fopen(targetFile, "wt")) == NULL)
{
  fprintf(stderr, "Cannot open output file \ file \n");
            exit(1);
}

            g_value = 0;
            pie = Pie();
            p = pjNO;
            q = rsNO;
            fprintf(out,"Source File = %s\n",sourceFile);
            fprintf(out,"Ray-Sum = %d\n",rsNO);
            fprintf(out,"Projections = %d\n",pjNO);
            fprintf(out,"Ray-Sum Interval = %5.4f\n",rayInterval);
            fprintf(out,"Angular Step = %5.4f\n",angularStep);
            minx = 1000; maxx = -1000;


            for(i=0; i< q; i++)          *
            {
                        percent = (i*100)/q ;
                        cprintf("%i %\r",percent);
                        for(j=0; j< q; j++)
                        {
                                    for(k=0;k<p; k++)
                                    {
                                                if( incycle(i,j) )
                                                {
                                                            ang
= angle(k);

r_point = rpoint(i,j,ang);

dif_r = r_point - floor(r_point);

if(dif_r > 0.00)

                                                            {

             point_lower = floor(r_point);

             point_upper = point_lower + 1;

             s_value = (ct[k][point_lower]+((ct[k][point_upper] -
ct[k][point_lower])*dif_r))/pie;
                                                            }
                                                            else
                                                            {

             s_value = ct[k][r_point]/pie;

                                                            }
                                                  g_value +=
s_value;
                                                }/* end if incycle */
                                                else
                                                {
                                                  g_value = -
1000.00;
                                                  k = p;
                                                }; /* end if cycle */
                                    }/* end k*/
                        fprintf(out,"%5.4f\n",g_value);
                        if ( (g_value < minx) && (g_value != -1000) ) minx =
g_value;

                        if (g_value > maxx) maxx = g_value;
                        g_value = 0;
                        } /* end j */
            }/*  end i */
            if (minAll > 999) minAll = minx;
            if (maxAll < -999) maxAll = maxx;
            minAll = min(minAll,minx);
            maxAll = max(maxx,maxAll);
            fprintf(out,"Minimum value = %5.4f\n",minx);
            fprintf(out,"Maximum value = %5.4f\n",maxx);
fclose(out);
}
//void readSource(void)
//{
//}
void readSource(void)
{
FILE *in;
long i = 0;
long j= 0;
int x;
char temp[80];
long nsize;



if ((in = fopen(sourceFile, "rt"))
   == NULL)
{
  fprintf(stderr, "Cannot open input \
            file.\n");
            exit(1);
}

            for(k=0;k<5;k++)
            {
                                    for (i = 0; i<250; i++){
                                                temp[i] = fgetc(in);
                                                if (temp[i] == '\n' ) {j
= i;i = 250;}
                                                            }
                                                temp[j] = '\0';
                        DOCOMMAND(temp);
            }

No_Profile = pjNO;
No_Raysum  = rsNO;
AngularStep = angularStep;
Ra_Interval = rayInterval;

j = rsNO*pjNO;
i=0;
            for(i=0;i<j;i++){
                                    fscanf(in,"%s",temp);
                                    fp[i] = atof(temp);
                                    }
```

```
fclose(in);
}


void ReadSource(unsigned long n_byte)
{
FILE *in;
int i = 0;
char temp[80];
long nsize;

if ((in = fopen(sourceFile, "rt"))
   == NULL)
{
  fprintf(stderr, "Cannot open input \
                  file.\n");
                  exit(1);
}
                if(n_byte > 15359) exit(1);
                while(!feof(in)){
                                        fscanf(in,"%s",temp);
                                        fp[i] = atof(temp);
                                        i++;
                                        }

fclose(in);
}



void readSection(int Sec)
{
unsigned long len;
struct PCXHEAD header;
unsigned char  pic[3000];
char tmp[25];
char *p;
FILE *in,*out;
int i,j,k,m;
unsigned int tt;
unsigned long l=0;
unsigned long ll;
     l=0;
     if ((PicRight-PicLeft)<No_Raysum){

No_Raysum = PicRight-PicLeft;
                                                            tt =
No_Raysum%2;

if(tt){ No_Raysum--; PicRight--;};
                                                            };
     if ((PicBottom-PicTop)<No_Section) No_Section = PicBottom -PicTop;
     if ((Sec > No_Section)&&(No_Section != 0)) Sec = No_Section;
     k = (PicRight+3-PicLeft)/No_Raysum;
//    No_Section += 1;
     if(No_Section > 2)
                    PicTC = PicTop + ((PicBottom-PicTop)/(No_Section-1))*Sec;
     else
                    PicTC = PicTop + ((PicBottom-PicTop)/(No_Section))*Sec;
for(m=0;m<No_Profile;m++)
{
     strcpy(sourceFile,NProfile[m]);
     if((in = fopen(sourceFile,"rb")) == NULL){
                exit(0);};

                  fread((char *)&header,1,sizeof(header),in);

                  for(i=0;i<PicTC;i++)
                                  ReadPcxLine(pic,in,header.bytes_per_line);

                  ReadPcxLine(pic,in,header.bytes_per_line);
                                  for(j=PicLeft;j<PicRight;j=j+k)
                                  {
                                   fp[l] = pic[j];
                                   l++;
                                  }
      fclose(in);
}
   strcpy(tmp,NSection[0]);
   p =  strtok(tmp,"0");
   if(p)    sprintf(sourceFile,"%s%i.DAT",p,Sec);
   else  { p = strtok(tmp,"."); sprintf(sourceFile,"%s%i.DAT",p,Sec);}

                if ((out = fopen(sourceFile, "wt"))    == NULL)
                {
                  fprintf(stderr, "Cannot open output file.\n");
                }
                fprintf(out,"Source File = %s\n",sourceFile);
                fprintf(out,"Ray-Sum = %d\n",rsNO);
                fprintf(out,"Projections = %d\n",pjNO);
                fprintf(out,"Ray-Sum Interval = %f\n",rayInterval);
                fprintf(out,"Angular Step = %f\n",angularStep);

                for(ll=0;ll<l;ll++){
                                                tt = fp[ll];
                                                fprintf(out,"%d\n",tt);}
                fclose(out);
}



void corn_tar(int s)
{
FILE *out;
int i,j,x;
long q;
  pjNO = No_Profile;
  rsNO = No_Raysum;
  rayInterval = Rs_Interval;
  angularStep = AngularStep;
  if (FormFile == 0) No_Section += 1;
  minAll = 1000;
  maxAll = -1000;
//              if ((fp = calloc(No_Profile*No_Raysum,sizeof(float))) == NULL)
//              {
//                printf("\nNo enough space exists for the float block\n");
//                exit(1);
//              }
clrscr();
  for(i=0;i<No_Section;i++)
  {
                if (FormFile == 0)
                {
                                                readSection(i);
                                                rsNO = No_Raysum;
                }
else{
                strcpy(sourceFile,NProfile[i]);
                readSource();
  }
switch(s){
        case 1:
                SheppLogan();
                break;
        case 2:
                RamLak();
                break;
        case 3:
                Convolution();
                                break;
                        default: break;
                        }
                printf("\r\nConvolution section %i of %i success\n",i,No_Section-1);
                strcpy(targetFile,NSection[i]);
                backprojection();
                printf("\r\nReconstruction section %i of %i success\n",i,No_Section-1);
//              tp = realloc(tp,No_Profile*No_Raysum);
  }
//  free(tp);
  SaveData();
  printf("\r\nSuccess Reconstruction\n");
}
```

SOURCE CODE OF EDIT.H

```
//extern time_t getEVEN(unsigned  *even);
//extern struct POSITION CurPosition;

char Filename[255];
char TEXT[200][82];

void SaveDat1(int fin)
{
FILE *wrt;
int i,j,k;
   if((wrt = fopen(Filename,"wt")) == NULL) exit(1);
   for(i=0;i<=fin;i++) fprintf(wrt,"%s\n",TEXT[i]);
   fclose(wrt);
   i = 13;j=0;k=0;
   do{ strcpy(NProfile[j],TEXT[i]);i++;j++;
   }while(TEXT[i][0] != '-');
   do{ i++;strcpy(NSection[k],TEXT[i]);k++;
   }while(TEXT[i+1][0] != '-');
}


void edit(char *name)
{
int i,j,tmp,quit=0;
int IdxC,IdxF;
int even;
FILE *in,*out;

 strcpy(Filename,name);
 if ((in = fopen(Filename, "rt")) == NULL) {IdxF=0;}
 else{
              i=0;j=0;
               do{
                 tmp=fgetc(in);
                 switch(tmp){
                                        case '\n':
                                        case '\r':
                                                    TEXT[i][j] ='\0';
                                                    i++;j=0;
                                                    break;
                                        default:
                                                    TEXT[i][j] = tmp;
                                                    j++;
                                                    break;
                                        }
               }while(!feof(in));
               IdxF=i-1;
   }
 fclose(in);
 clrscr();
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqq\n");
for(i=0;i<23;i++)
printf("ßı                                          ßı\n");
printf("#qqqq<ESC>EXITqNOTqCHANGEqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
<F2>EXITqANDqSAVEqqqqqqqqqq");
              tmp = IdxF;
              if(tmp > 22) tmp =22;
              IdxC=0;
                        gotoxy(2,2);
                        for(i=0;i<=tmp;i++){ printf("%s",TEXT[i]);
                                                   gotoxy(2,i+3);
                                                   }
      gotoxy(2,2);
      do{
                        getEVEN(&even);
                        switch(even){
                                                   case 0: break;
                                                   case K_HOME:
gotoxy(2,CurPosition.y); break;
                                                   case K_LEFT: if(CurPosition.x>2)

                        gotoxy(CurPosition.x-1,CurPosition.y);break;
                                                   case K_RIGHT: if(CurPosition.x<79)

                        gotoxy(CurPosition.x+1,CurPosition.y);break;
                                                   case K_UP:   IdxC--;
                                                               if(CurPosition.y >
2) gotoxy(CurPosition.x,CurPosition.y-1);
                                                               if((CurPosition.y
== 2)&&(IdxC>=0))
                                                               {

movetext(2,2,78,23,2,3);

gotoxy(2,2);clreol();

printf("%s",TEXT[IdxC]);

gotoxy(79,2); printf("ßı");

gotoxy(CurPosition.x,2);

                                                               }
                                                               if(IdxC<0) IdxC
= 0;
                                                               break;
                                                   case K_DOWN: IdxC++;

if(CurPosition.y<24)
                        gotoxy(CurPosition.x,CurPosition.y+1);
                                                               if((CurPosition.y
== 24)&&(IdxC<=IdxF))
                                                                           {

movetext(2,3,78,24,2,2);

gotoxy(2,24);clreol();

printf("%s",TEXT[IdxC]);

gotoxy(79,24); printf("ßı");

gotoxy(CurPosition.x,24);

                                                                           }
                                                               if(IdxC>IdxF)
IdxC = IdxF;
                                                               break;
                                                   case K_F2: SaveDat1(IdxF);
```

```
                                    case K_ESC: quit = 1; break;
                                    default:

                TEXT[InxC][CurPosition.x-2] = even;

                TEXT[InxC][CurPosition.x-1] = '\0';
                                                    putch(even);
                                                    break;
                              }
     }while(quit==0);
}

/*PCX Read*/
/*struct PCXHEAD {
                              char manufacturer;
                              char version;
                              char encoding;
                              char bit_per_pixel;
                              int xmin,ymin;
                              int xmax,ymax;
                              int hres;
                              int vres;
                              char palette[48];
                              char reserved;
                              char colour_planes;
                              int bytes_per_line;
                              int palette_type;
                              char filler[58];
                    };
*/
int ReadPcxLine(char *p,FILE *fp,unsigned int bytes)
{
int n=0,c,i;
   memset(p,0,bytes);
   do{
     c=fgetc(fp) & 0xff;
     if((c&0xc0) == 0xc0)
     {
               i=c & 0x3f;
               c=fgetc(fp);
               while(i--) p[n++]=c;
     }else p[n++]=c;
   }while(n<bytes);
   return n;
}

void cleargr(int L,int T, int R, int B)
{
 setviewport(L,T,R,B,1);
 clearviewport();
 setviewport(0,0,639,479,1);
}

void SetVGApalette(char *p)
{
int i,r,g,b;
  for(i=1;i<769;i++) p[i] = p[i] >>2;
  for(i=1;i<769;i+=3){
                                        r = p[i]; g = p[i+1]; b = p[i+2];
                                        setrgbpalette(i/3,r,g,b);
                              }
}

void save_line(int x1,int y1,int x2,int y2,void far *buf[4])
{
  unsigned size;
  int block;

  size = imagesize(x1, y1, x2, y2);
/* get byte size of image */

  for (block=0; block<=3; block++)
  {
    if ((buf[block] = farmalloc(size)) == NULL)
    {
               closegraph();
               printf("Error: not enough heap space in save_screen().\n");
               exit(1);
    }

    getimage(x1, y1, x2, y2, buf[block]);
  }
}

void restore_line(int x1,int y1,int opt,void far *buf[4])
{
  int  block;


  for (block=0; block<=3; block++)
  {
    putimage(x1, y1, buf[block], opt);
    farfree(buf[block]);
  }
}


void Grfinput(void)
{
int k,ratioX,ratioY;
int i,a,r,t,b,z;
unsigned color,bkcol,color1;
struct PCXHEAD header;
int gdriver = VGA, gmode = VGAHI, errorcode;
char pic[3000];
char pic2[3000];
char palette[769];
FILE *fpf;
unsigned int even;
int quit = 1;
int Inx;
void far *ptr[4];

//strcpy(NProfile[0],"THE_TEST.PCX");
if((fpf=fopen(NProfile[0],"rb")) != NULL){
                                                                 fread((char
*)&header,1,sizeof(header),fpf);
                                                                 fseek(fpf,-
769L,SEEK_END);

fread(palette,1,769,fpf);

fseek(fpf,128L,SEEK_SET);
                                        }
gdriver = installuserdriver("SVGA256", Detect);
gmode = 2;
```

```
initgraph(&gdriver, &gmode, "");
errorcode = graphresult();

if (errorcode != grOk)
{
  printf("Graphics error: %s\n", grapherrormsg(errorcode));
  printf("Press any key to halt:");
  exit(1);
}
                if(palette[0] == 0x0c)
                  SetVGApalette(palette);
//                for(i=0;i<255;i++) {putpixel(i,i,i);}
                PicMaxX = (header.xmax-header.xmin)+1;
                ratioX = 1; ratioY =1;
                if(PicMaxX > 255)
                  ratioX = PicMaxX/256 + 1;
                PicMaxY = (header.ymax-header.ymin)+1;
                if(PicMaxY > 255)
                  ratioY = PicMaxY/256 + 1;
                PicLeft = 0; PicRight = PicMaxX;
                PicTop = 0; PicBottom = PicMaxY;
                block(0,0,639,479);
                for(i=0;i<PicMaxY;i++){

ReadPcxLine(pic,fpf,header.bytes_per_line);
                                                if((k=fmod(i,ratioY))==0 )
                                                {

for(n=0;n<PicMaxX;n++)
                                                {

if((k=fmod(n,ratioX))==0)  putpixel(380 + n/ratioX,50+i/ratioY,pic[n]);
                                                }
                                                }
                                        }
    line(1,430,638,430);
    color = getpixel(5,440);
    setcolor(color^0xffff);
    outtextxy(5,440,"<ESC> to MAIN MENU");
//    outtextxy(5,460,"<D> to DEFAULT");
    outtextxy(250,440,"<L> to set LEFT");
    outtextxy(400,440,"<R> to set RIGHT");
    outtextxy(250,460,"<T> to set TOP");
    outtextxy(400,460,"<B> to set BOTTOM");
//    setcolor(4);
    k = 155;
    sprintf(pic,"NO. OF PROFILES = %i",No_Profile);
    outtextxy(30,100+k,pic);
    sprintf(pic,"ANGULAR STEP = %3.2f",AngularStep);
    outtextxy(30,115+k,pic);
    sprintf(pic,"NO. OF RAY-SUM = %i",No_Raysum);
    outtextxy(30,130+k,pic);
    sprintf(pic,"RAY-SUM INTERVAL = %3.2f",Rs_Interval);
    outtextxy(30,145+k,pic);
    sprintf(pic,"NO. OF SECTIONS = %i",No_Section);
    outtextxy(30,160+k,pic);
    sprintf(pic,"SECTIONS INTERVAL = %3.2f",Sec_Interval);
    outtextxy(30,175+k,pic);
    sprintf(pic,"MAXIMUM (X) = %i",PicMaxX);
    outtextxy(30,190+k,pic);
    sprintf(pic,"MAXIMUM (Y) = %i",PicMaxY);
    outtextxy(30,205+k,pic);
    sprintf(pic,"LEFT   = %i",PicLeft);
    outtextxy(30,220+k,pic);
    sprintf(pic,"TOP    = %i",PicTop);
    outtextxy(30,235+k,pic);
    sprintf(pic,"RIGHT  = %i",PicRight);
    outtextxy(30,250+k,pic);
    sprintf(pic,"BOTTOM = %i",PicBottom);
    outtextxy(30,265+k,pic);
    color = getpixel(15,230);
    setcolor(color^0xffff);
    block(15,250,230,430);
fclose(fpf);
    setrgbpalette(4,32,32,32);
    bkcol = getpixel(10,10);
    for(i=25;i<250;i+=25) { line(25,248-i,29,248-i);
                                        line(30+i,240,30+i,248);
                                }

    k=0; r=0, t =0,b=0;
    do{
            getEVEN(&even);
            switch(even){
                                        case 9804: /*L*/
                                        case 9836: Inx = 1;
                                                        color =
getpixel(10,20);
clearvp(470,345,630,365);
                                                        sprintf(pic,"LEFT
= %i",PicLeft);
setcolor(bkcol^0xffff);
outtextxy(489,350,pic);
                                                        break;
                                        case 4978: /*R*/
                                        case 4946: Inx = 2;
                                                        color =
getpixel(10,20);
clearvp(470,345,630,365);
                                                        sprintf(pic,"RIGHT
= %i",PicRight);
setcolor(bkcol^0xffff);
outtextxy(489,350,pic);
                                                        break;
                                        case 5236:/*T*/
                                        case 5204: Inx = 3;
                                                        color =
getpixel(10,20);
clearvp(470,345,630,365);
                                                        sprintf(pic,"TOP =
%i",PicTop);
setcolor(bkcol^0xffff);
outtextxy(489,350,pic);
                                                        break;
                                        case 12386:/*B*/
                                        case 12354: Inx =4;
                                                        color =
getpixel(10,20);
```

```
cleargr(470,345,630,365);

sprintf(pic,"BOTTOM = %i",PicBottom);

setcolor(bkcol^0xffff);

outtextxy(489,350,pic);
                                                            break;
                                              case 18989:
                                              case 3117:
                                                            switch(lnx)
                                                            {
                                                              case 1:
setcolor(bkcol);

if(k <= 1) k = 1;

line(380+k,10,380+k,49);

line(380+k,52+PicMaxY/ratioY,380+k,80+PicMaxY/ratioY);

sprintf(pic,"LEFT = %i",PicLeft);

outtextxy(489,350,pic);
//
if(k>0) putpixel(380+k,52+(PicMaxY/(2*ratioY)),color1);

k--;

if(k>0){
              for(i=0;i<PicMaxY;i++)

              {

               color1 = getpixel(380+k,50+i);

               putpixel(380+k,50+i,(color1^0xFF));

               color1 = getpixel(381+k,50+i);

               putpixel(381+k,50+i,(color1^0xFF));

              }

             }
setcolor(bkcol^0xffff);

line(380+k,10,380+k,49);

line(380+k,52+PicMaxY/ratioY,380+k,80+PicMaxY/ratioY);

PicLeft = k *ratioX;

color = getpixel(10,20);

setcolor(color^0xffff);

sprintf(pic,"LEFT = %i",PicLeft);

outtextxy(489,350,pic);
```

```
break;
                                                              case 2:
setcolor(bkcol);

line(380-r+PicMaxX/ratioX,10,380-r+PicMaxX/ratioX,49);

line(380-r+PicMaxX/ratioX,52+PicMaxY/ratioY,380-
r+PicMaxX/ratioX,80+PicMaxY/ratioY);

sprintf(pic,"RIGHT = %i",PicRight);

outtextxy(489,350,pic);
//
if(r<PicMaxX) putpixel(380-r+PicMaxX/ratioX,52+(PicMaxY/(2*ratioY)),color1);

r++;

if(r<PicMaxX){

                          for(i=0;i<PicMaxY;i++)

                          {

                           color1 = getpixel(380-r+PicMaxX/ratioX,50+i);

                           putpixel(380-
r+PicMaxX/ratioX,50+i,(color1^0xFF));

                           color1 = getpixel(381-r+PicMaxX/ratioX,50+i);

                           putpixel(381-
r+PicMaxX/ratioX,50+i,(color1^0xFF));

                          }

                          }

setcolor(bkcol^0xffff);

line(380-r+PicMaxX/ratioX,52+PicMaxY/ratioY,380-
r+PicMaxX/ratioX,80+PicMaxY/ratioY);

line(380-r+PicMaxX/ratioX,10,380-r+PicMaxX/ratioX,49);

PicRight = PicMaxX-(r * ratioX);

color = getpixel(10,20);

setcolor(color^0xffff);

sprintf(pic,"RIGHT = %i",PicRight);

outtextxy(489,350,pic);

break;
                                                              case 3:
setcolor(bkcol);

if(t<1) t=1;

line(355,50+t,379,50+t);
/*
save_line(380,50+t,380+255,50+t);
```

```
*/
sprintf(pic,"TOP = %i",PicTop);

outtextxy(489,350,pic);
//
if(t>0) putpixel(380+PicMaxX/(2*ratioX),50+t,color1);

for(i=1;i<241;i++)  line(30,i,30+PicMaxX,i);

if(t>0){

                for(i=0;i<PicMaxX;i++)

                {

                 color1 = getpixel(380+i,50+t);

                 putpixel(380+i,50+t,(color1^0xFF));

                }

                }

t--;

if(t>0){

                for(i=0;i<PicMaxX;i++)

                {

                 color1 = getpixel(380+i,50+t);

                 putpixel(380+i,50+t,(color1^0xFF));

                 putpixel(30+i,240-color1*.8,bkcol^0xffff);

                }

                }

setcolor(bkcol^0xffff);

line(355,50+t,379,50+t);

PicTop = (t * ratioY);

color = getpixel(10,20);

setcolor(color^0xffff);

sprintf(pic,"TOP = %i",PicTop);

outtextxy(489,350,pic);

break;
                                                case 4:
setcolor(bkcol);

if(b<(-1*PicMaxY)) b = -1*PicMaxY;

line(355,50+(PicMaxY/ratioY)+b,379,50+(PicMaxY/ratioY)+b);

sprintf(pic,"BOTTOM = %i",PicBottom);
```

```
outtextxy(489,350,pic);
//
if(b<0) putpixel(380+PicMaxX/(2*ratioX),50+(PicMaxY/ratioY)+b,color1);

for(i=1;i<241;i++)  line(30,i,30+PicMaxX,i);

if(b<0){

                for(i=0;i<PicMaxX;i++)

                {

                 color1 = getpixel(380+i,50+(PicMaxY/ratioY)+b);

                 putpixel(380+i,50+(PicMaxY/ratioY)+b,(color1^0xFF));

                }

                }

b--;

if(b<0){

                for(i=0;i<PicMaxX;i++)

                {

                 color1 = getpixel(380+i,50+(PicMaxY/ratioY)+b);

                 putpixel(380+i,50+(PicMaxY/ratioY)+b,(color1^0xFF));

                 putpixel(30+i,240-color1*.8,bkcol^0xffff);

                }

                }

setcolor(bkcol^0xffff);

line(355,50+(PicMaxY/ratioY)+b,379,50+(PicMaxY/ratioY)+b);

PicBottom = PicMaxY+(b * ratioY);

color = getpixel(10,20);

setcolor(color^0xffff);

sprintf(pic,"BOTTOM = %i",PicBottom);

outtextxy(489,350,pic);

break;
                                                              default: break;
                                                             }
                                                             break;
                                                case K_PLUS:
                                                case 3371:
                                                             switch(Imx)
                                                             {
                                                              case 1:
setcolor(bkcol);
```

```
line(380+k,10,380+k,49);

line(380+k,52+PicMaxY/ratioY,380+k,80+PicMaxY/ratioY);

sprintf(pic,"LEFT = %i",PicLeft);

outtextxy(489,350,pic);
//
if(k>0) putpixel(380+k,52+(PicMaxY/(2*ratioY)),color1);

k++;

if(k>0){

                for(i=0;i<PicMaxY;i++)

                {

                  color1 = getpixel(380+k,50+i);

                  putpixel(380+k,50+i,(color1^0xFF));

                  color1 = getpixel(379+k,50+i);

                  putpixel(379+k,50+i,(color1^0xFF));

                }

                }

color = getpixel(10,20);

setcolor(color^0xffff);

line(380+k,52+PicMaxY/ratioY,380+k,80+PicMaxY/ratioY);

line(380+k,10,380+k,49);

PicLeft = k * ratioX;

setcolor(color^0xffff);

sprintf(pic,"LEFT = %i",PicLeft);

outtextxy(489,350,pic);

break;
                                                            case 2:
setcolor(bkcol);

if(r <= 1) r=1;

line(380-r+PicMaxX/ratioX,10,380-r+PicMaxX/ratioX,49);

line(380-r+PicMaxX/ratioX,52+PicMaxY/ratioY,380-
r+PicMaxX/ratioX,80+PicMaxY/ratioY);

sprintf(pic,"RIGHT = %i",PicRight);

outtextxy(489,350,pic);
//
if(r>0) putpixel(380-r+PicMaxX/ratioX,52+(PicMaxY/(2*ratioY)),color1);
```

```
r--;

if(r>0){

                            for(i=0;i<PicMaxY;i++)

                            {

                             color1 = getpixel(380-r+PicMaxX/ratioX,50+i);

                             putpixel(380-
r+PicMaxX/ratioX,50+i,(color1^0xFF));

                             color1 = getpixel(379-r+PicMaxX/ratioX,50+i);

                             putpixel(379-
r+PicMaxX/ratioX,50+i,(color1^0xFF));

                            }

                    }

color = getpixel(10,20);

setcolor(color^0xffff);

line(380-r+PicMaxX/ratioX,52+PicMaxY/ratioY,380-
r+PicMaxX/ratioX,80+PicMaxY/ratioY);

line(380-r+PicMaxX/ratioX,10,380-r+PicMaxX/ratioX,49);

PicRight = PicMaxX-(r * ratioX);

setcolor(color^0xffff);

sprintf(pic,"RIGHT = %i",PicRight);

outtextxy(489,350,pic);

break;
                                                            case 3:
setcolor(bkcol);

if(t>PicMaxY) t = PicMaxY;

line(355,50+t,379,50+t);

sprintf(pic,"TOP = %i",PicTop);

outtextxy(489,350,pic);
//
if(t>0) putpixel(380+PicMaxX/(2*ratioX),50+t,color1);

setcolor(bkcol);

for(i=1;i<241;i++)  line(30,i,30+PicMaxX,i);

if(t>0){

                for(i=0;i<PicMaxX;i++)

                {
```

```
                color1 = getpixel(380+i,50+t);
                putpixel(380+i,50+t,(color1^0xFF));
                }
               }
t++;
if(t>0){
               for(i=0;i<PicMaxX;i++)
               {
               color1 = getpixel(380+i,50+t);
               putpixel(380+i,50+t,(color1^0xFF));
               putpixel(30+i,240-color1*.8,bkcol^0xffff);
               }
              }
color = getpixel(10,20);
setcolor(color^0xffff);
line(355,50+t,379,50+t);
PicTop = (t * ratioY);
setcolor(color^0xffff);
sprintf(pic,"TOP = %i",PicTop);
outtextxy(489,350,pic);
break;
                                    case 4:
setcolor(bkcol);
if(b>-1) b = -1;
line(355,50+(PicMaxY/ratioY)+b,379,50+(PicMaxY/ratioY)+b);
sprintf(pic,"BOTTOM = %i",PicBottom);
outtextxy(489,350,pic);
//
if(b<0) putpixel(380+PicMaxX/(2*ratioX),50+(PicMaxY/ratioY)+b,color1);
for(i=1;i<241;i++)  line(30,i,30+PicMaxX,i);
if(b<0){
              for(i=0;i<PicMaxX;i++)
              {
              color1 = getpixel(380+i,50+(PicMaxY/ratioY)+b);
```

```
              putpixel(380+i,50+(PicMaxY/ratioY)+b,(color1^0xFF));
              }
             }
b++;
if(b<0){
              for(i=0;i<PicMaxX;i++)
              {
              color1 = getpixel(380+i,50+(PicMaxY/ratioY)+b);
              putpixel(380+i,50+(PicMaxY/ratioY)+b,(color1^0xFF));
              putpixel(30+i,240-color1*.8,bkcol^0xffff);
              }
             }
color = getpixel(10,20);
setcolor(color^0xffff);
line(355,50+(PicMaxY/ratioY)+b,379,50+(PicMaxY/ratioY)+b);
PicBottom = PicMaxY+(b * ratioY);
setcolor(color^0xffff);
sprintf(pic,"BOTTOM = %i",PicBottom);
outtextxy(489,350,pic);
break;
                                          default: break;
                                         }
                                         break;
                                  case K_ESC: quit = !quit; break;
                                  case K_F1:
//                                conv_tur();
//                                for(z=0;z<No_Raysum;z++)
putpixel(z,50,(int) fp[z]);
                                  case 0: break;
                                  default: break;
                      }
      }while(quit);
closegraph();
}


int Pcx256Type(void)
{
int ret;
FILE *fp;
struct PCXHEAD header;
char palette[769];
int type;
```

```
if((fpf=fopen(NProfile[0],"rb")) != NULL){

*)&header,1,sizeof(header),fpf);

769L,SEEK_END);

fread(palette,1,769,fpf);

fread((char

fseek(fpf,-

fseek(fpf,128L,SEEK_SET);

return 1;
}

}
```

SOURCE CODE OF GR3D.H

```
#define PiOver180   1.74532925199433E-002
#define PiUnder180  5.72957795130823E+001

void Showprofile(void);
void PALETTE(void);
void gr3dl(void);

int vTop = 100;
int vBottom = 305;
int vLeft = 150;
int vRight = 305;

int BW64(int indx)
{
int i,j,k;
                j = indx*64;
                k = j+64;
                for(i=j;i<k;i++) setrgbpalette(i,i,i,i);
                return j;
}
/*
                start at 64 to  192
*/
void BW128(void)
{
int i;
    for(i=0;i<64;i++) setrgbpalette(64+(i*2),i,i,i);
    for(i=0;i<64;i++) setrgbpalette(65+(i*2),i+1,i,i);
}



/*
                start at 64 to  255
*/
void BW192(void)
{
int i;
    for(i=0;i<64;i++) setrgbpalette(64+(i*3),i,i,i);
    for(i=0;i<63;i++) setrgbpalette(65+(i*3),i+1,i,i);
    for(i=0;i<63;i++) setrgbpalette(66+(i*3),i,i+1,i);
   setrgbpalette(65+(i*3),i,i,i);
   setrgbpalette(66+(i*3),i,i,i);
}

int BW64R(int indx)
{
int i,j,k;
                j = indx*64;
                k = j+64;
                for(i=j;i<k;i++) setrgbpalette(k-i,i,i,i);
                return j;
}

void BW128R(void)
{
int i;
    for(i=0;i<64;i++) setrgbpalette(191-(i*2),i,i,i);
    for(i=0;i<64;i++) setrgbpalette(190-(i*2),i+1,i,i);
}
```

```
void BW192R(void)
{
int i;
    for(i=0;i<64;i++) setrgbpalette(253-(i*3),i,i,i);
    for(i=0;i<63;i++) setrgbpalette(254-(i*3),i,i+1,i);
    for(i=0;i<63;i++) setrgbpalette(255-(i*3),i+1,i,i);
    setrgbpalette(254-(i*3),63,63,63);
    setrgbpalette(255-(i*3),63,63,63);


}

int showprofile(void)
{
int i,j,q = 1;
char tmp[20];

gr3dl();
do {
clrscr();
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqq\n");
for(i=0;i<23;i++)
printf("3:                                                  3\n");
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqq");
  for(i=0;i<No_Profile;i++)
   {
                if(i<20){
                                gotoxy(5,2+i);printf("%2i - %s",i,NProfile[i]);
                                }
                if((i>19)&&(i<40)){
                                                gotoxy(35,i-18);printf("%i -
%s",i,NProfile[i]);
                                                }
                if((i>39)&&(i<60)){
                                                gotoxy(62,i-38);printf("%i -
%s",i,NProfile[i]);
                                                }
   }
gotoxy(2,22);
printf("<q> to quit-----------------------------------------------");
gotoxy(30,23);printf("SELECT NUMBER -> ");scanf("%s",tmp);

 if((tmp[0] != 'q') && (tmp[0] != 'Q')) {
                                                                        j =
atoi(tmp);

                strcpy(targetFile,NProfile[j]);

                Showprofile();

                }else {q = !q;}
}while(q);
return 1;
}


void Showprofile(void)
{
int k,ratioX,ratioY;
int i,n,r,t,h,z;
```

```
unsigned color,bkcol;
struct PCXHEAD header;
int gdriver = VGA, gmode = VGAHI, errorcode;
char pic[3000];
char palette[769];
char c;
int cc;
unsigned strock[60][255];
FILE *fpf;
unsigned int even;
int quit = 1;
int lnx;
//union REGS rr;
//strcpy(NProfile[0],"THE_TEST.PCX");
if((fpf=fopen(targetFile,"rb")) != NULL){
                                                fread((char
*)&header,1,sizeof(header),fpf);
                                                if(!fseek(fpf,-769L,SEEK_END));
                                                lnx = fread(palette,1,769,fpf);
                                                if(!fseek(fpf,128L,SEEK_SET));
                                                }
   lnx = header.bit_per_pixel;
gdriver = installuserdriver("SVGA256", Detect);
gmode = 2;
initgraph(&gdriver, &gmode, "");
errorcode = graphresult();

if (errorcode != grOk)
{
  printf("Graphics error: %s\n", grapherrormsg(errorcode));
  printf("Press any key to halt:");
  exit(1);
}
                if(palette[0] == 0x0c)
                    SetVGApalette(palette);
//              BW192();
                PicMaxX = (header.xmax-header.xmin)+1;
                ratioX = 1; ratioY =1;
                if(PicMaxX > 255)
                  ratioX = PicMaxX/256 + 1;
                PicMaxY = (header.ymax-header.ymin)+1;
                if(PicMaxY > 255)
                  ratioY = PicMaxY/256 + 1;
//              PicLeft = 0; PicRight = PicMaxX;
//              PicTop = 0; PicBottom = PicMaxY;
                block(0,0,639,479);
                r =-1;
//      PicTC = PicTop + ((PicBottom-PicTop)/(No_Section-2))*Sec;
//              t = (PicBottom - PicTop)/(No_Section-2) ;
                for(i=0;i<PicMaxY;i++){
ReadPcxLine(pic,fpf,header.bytes_per_line);
                                                if((k=fmod(i,ratioY))==0 )
                                                      {  if(((z=fmod(abs(i-
PicTop),t)) == 0) &&(i>=PicTop) && (i<=PicBottom)) r++;

for(n=0;n<PicMaxX;n++)
                                                      {

if((b=fmod(n,ratioX))==0) {

                                    putpixel(380 + n/ratioX,50+i/ratioY,(pic[n]));

                              }
```

```
                                                            }
                                                          }
                                                      }
      line(1,430,638,430);
      color = getpixel(5,440);
      setcolor(color^0xffff);
      outtextxy(5,440,"<ESC> to MAIN MENU");
//     outtextxy(5,460,"<K_DOWN> to DEFAULT");
      outtextxy(250,440,"<K_DOWN> to next section");
//     outtextxy(400,440,"<K_UP> to set RIGHT");
      outtextxy(250,460,"<K_UP> to previous section");
//     outtextxy(400,460,"<B> to set BOTTOM");
      color = getpixel(380,330);
      setcolor(color^0xffff);
      sprintf(pic,"Profile : %s",targetFile);
      outtextxy(380,330,pic);
      line(29,40,29,316);
      outtextxy(25,35,"x25");
      outtextxy(288,316,"x25");
      for(i=25;i<275;i+=25) { line(25,316-i,29,316-i);
                                                line(30+i,316,30+i,321);
                                              }
//     line(29,316,285,316);
//     PicTC = PicTop + ((PicBottom-PicTop)/(No_Section-2))*Sec;
      b=0;
      bkcol = getpixel(31,315);
      for(k=0;k<No_Section;k++)
      {
                  t = ((PicBottom-PicTop)/(No_Section-1))*k;
                  for(r=PicLeft;r<PicRight;r++){

                strock[b][r/ratioX] = getpixel(380 + r/ratioX,50+PicTop/ratioY+t);
                                                            }
                  b++;
      }
      cc = ((float)(PicBottom-PicTop)/(float)(No_Section-1));
      k=0; r=0, t =0,b=0;
//     for(k=PicTop;k<PicBottom;k++)
                  for(i=PicLeft;i<PicRight;i++){

putpixel(30+i,315-strock[b][i],bkcol^0xffff);
                                                            }
                  line(370,50+PicTop/ratioY,379,50+PicTop/ratioY);
      color = getpixel(100,325);
      setcolor(color^0xffff);
      sprintf(pic,"Section %i",b);
      outtextxy(100,325,pic);
      do{
                  getEVEN(&even);
                  switch(even){
                                                    case K_DOWN:
                                                        setcolor(bkcol);
                                                        for(k=PicTop;k<PicBottom;k++)

for(i=PicLeft;i<PicRight;i++)

                  putpixel(30+i,315-strock[b][i],bkcol);

sprintf(pic,"Section %i",b);

outtextxy(100,325,pic);
                                                                        cc =
((PicBottom-PicTop)/(No_Section-1))*b;
```

```
line(370,50+PicTop/ratioY+cc,379,50+PicTop/ratioY+cc);
                                                        b++;
if(b>=No_Section) b = No_Section-1;
                                                        color =
getpixel(100,325);

setcolor(color^0xffff);

sprintf(pic,"Section %i",b);

outtextxy(100,325,pic);
                                                        cc =
((PicBottom-PicTop)/(No_Section-1))*b;

line(370,50+PicTop/ratioY+cc,379,50+PicTop/ratioY+cc);
//                                              for(k=PicTop;k<PicBottom;k++)

for(i=PicLeft;i<PicRight;i++)

                putpixel(30+i,315-strock[b][i],color^0xffff);
                                                        break;
                                                case K_UP:
                                                    setcolor(bkcol);
                                                    for(k=PicTop;k<PicBottom;k++)

for(i=PicLeft;i<PicRight;i++)

                putpixel(30+i,315-strock[b][i],bkcol);

sprintf(pic,"Section %i",b);

outtextxy(100,325,pic);
                                                        cc =
((PicBottom-PicTop)/(No_Section-1))*b;

line(370,50+PicTop/ratioY+cc,379,50+PicTop/ratioY+cc);
                                                        b--; if(b<=0) b =
0;
//                                              for(k=PicTop;k<PicBottom;k++)
                                                        color =
getpixel(100,325);

for(i=PicLeft;i<PicRight;i++)

                putpixel(30+i,315-strock[b][i],color^0xffff);

setcolor(color^0xffff);
                                                        cc =
((PicBottom-PicTop)/(No_Section-1))*b;

line(370,50+PicTop/ratioY+cc,379,50+PicTop/ratioY+cc);

sprintf(pic,"Section %i",b);

outtextxy(100,325,pic);
                                                        break;
                                                case K_ESC: quit = !quit; break;
                                                case 0: break;
                                                default: break;
                                }
      }while(quit);
fclose(fp);
closegraph();
```

```
}




float A = 205,B = 15;

void Tran3to2(float x,float y,float z,float *Rx,float *Ry,float *Rz)
{
int Cx,Cy,Cz;
int Mx,My,Mz;
                Cx = (vLeft + vRight)/2;
                Cy = (vTop + vBottom)/2;
                Cz = 0;
//              Mx = -127; My = -127; Mz = -127;
//              x = x+Mx; y = y+My; z = z+Mz;
                *Rx = (Cx + x*cos(A*PiOver180) - y*sin(A*PiOver180));
                *Ry = (Cy - x*sin(A*PiOver180)*sin(B*PiOver180) +
y*cos(A*PiOver180)*sin(B*PiOver180) + z*cos(B*PiOver180));
                *Rz = (Cz + x*sin(A*PiOver180)*cos(B*PiOver180) +
y*cos(A*PiOver180)*cos(B*PiOver180) - z*sin(B*PiOver180));
}

float CosRotAngle;
float SinRotAngle;
void RotateX(float x,float y,float z,float *Rx,float *Ry,float *Rz)
{
  *Rx = x;
  *Ry = y*CosRotAngle + z*SinRotAngle;
  *Rz = -1*y*SinRotAngle + z*CosRotAngle;
}

void RotateY(float x,float y,float z,float *Rx,float *Ry,float *Rz)
{
  *Rx = x*CosRotAngle - z*SinRotAngle;
  *Ry = y;
  *Rz = x*SinRotAngle + z*CosRotAngle;
}

void RotateZ(float x,float y,float z,float *Rx,float *Ry,float *Rz)
{
  *Rx = x*CosRotAngle + y*SinRotAngle;
  *Ry = -1*x*SinRotAngle + y*CosRotAngle;
  *Rz = z;
}

void box(void)
{
float x,y,z;
int x1,x2,y1,y2;
                Tran3to2(-1,-1,-1,&x,&y,&z);    x1 = x; y1 = y;
                Tran3to2(256,-1,-1,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);
                Tran3to2(-1,256,-1,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);
                Tran3to2(-1,-1,256,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);
//
                Tran3to2(256,256,256,&x,&y,&z);  x1 = x; y1 = y;
                Tran3to2(256,256,-1,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);
                Tran3to2(-1,256,256,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);
                Tran3to2(256,-1,256,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);
//
                Tran3to2(-1,256,-1,&x,&y,&z);  x2 = x; y2 = y;
                Tran3to2(256,256,-1,&x,&y,&z);  x1 = x; y1 = y;
                line(x1,y1,x2,y2);
                Tran3to2(256,-1,-1,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);
//
                Tran3to2(-1,-1,256,&x,&y,&z);  x1 = x; y1 = y;
                Tran3to2(256,-1,256,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);
                Tran3to2(-1,256,256,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);
//
                Tran3to2(-1,256,0,&x,&y,&z);  x1 = x; y1 = y;
                Tran3to2(-1,256,256,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);
//
                Tran3to2(256,-1,-1,&x,&y,&z);  x1 = x; y1 = y;
                Tran3to2(256,-1,256,&x,&y,&z);  x2 = x; y2 = y;
                line(x1,y1,x2,y2);

}

void gr3dI(void)
{
FILE *in,*in2;
char temp[255];
int i,j;
struct palettetype pal;
fpos_t *pos;
float v,k;
int f,c;
unsigned even,q=1;

if(IDENPASS) goto PASS1;

clrscr();
printf("qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqqqw\n");
for(i=0;i<23;i++)
printf("x                                                   x\n");
printf("mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qqqqqqqqqqqqqqqqqqqqqqqqq");
gotoxy(2,2);cprintf("Thesis Title : DEVELOPMENT OF THREE DIMENSIONAL
COMPUTED TOMOGRAPHY SOFTWARE");
gotoxy(2,3);cprintf("By             : MR. CHERNGCHY SOYPETCH");
gotoxy(2,4);cprintf("Department       : NUCLEAR TECHNOLOGY");
gotoxy(2,5);cprintf("Thesis Advisor    : ASST. PROF. SOMYOT SRISATIT");
gotoxy(2,6);cprintf("Thesis Co-advisor : -");
gotoxy(2,7);cprintf("————————————————————————————————————————");
gotoxy(2,22);cprintf("————————————————————————————————————————");
gotoxy(10,23);cprintf("Copyright 1997 by Graduate School Chulalongkorn University.");
gotoxy(25,24);cprintf("All rights reserved worldwide.");
gotoxy(23,10);cprintf("INPUT IDENTIFIER FILE NAME SCREEN");

NEWIN1:;
gotoxy(24,14);cprintf("INPUT FILE NAME -: "); scanf("%s",NConfig);
if((strcmp(NConfig,"q") == 0) || (strcmp(NConfig,"Q") ==0)) goto PASS1;
                if ((in = fopen(NConfig, "rt"))    == NULL)
                  { gotoxy(23,17);cprintf("Cannot open input file.");
                   goto NEWIN1;
                                                }
                                                 for (i = 0; i<250; i++){
                                                                temp[i] = fgetc(in);
                                                                if (temp[i] == '\n' ) {j = i;i = 250;}
                                                                                }
                                                                temp[j] = '\0';
                                                 for (i = 0; i < strlen(temp); i++ ) {
                                                                switch (temp[i]) {
                                                                                case '=': temp[i] = ' ';i = strlen(temp);
                                                break;
                                                                                default: temp[i] = ' ';break;
                                                                                }
                                                                                                }
                                                lclrblank(temp);
                                                No_Profile = atoi(temp);
                                                 for (i = 0; i<250; i++){
                                                                temp[i] = fgetc(in);
                                                                if (temp[i] == '\n' ) {j = i;i = 250;}
                                                                                }
                                                                temp[j] = '\0';
                                                 for (i = 0; i < strlen(temp); i++ ) {
                                                                switch (temp[i]) {
                                                                                case '=': temp[i] = ' ';i = strlen(temp);
                                                break;
                                                                                default: temp[i] = ' ';break;
                                                                                }
                                                                                                }
                                                lclrblank(temp);
                                                No_Raysum = atoi(temp);
                                                 for (i = 0; i<250; i++){
                                                                temp[i] = fgetc(in);
                                                                if (temp[i] == '\n' ) {j = i;i = 250;}
                                                                                }
                                                                temp[j] = '\0';
                                                 for (i = 0; i < strlen(temp); i++ ) {
                                                                switch (temp[i]) {
                                                                                case '=': temp[i] = ' ';i = strlen(temp);
                                                break;
                                                                                default: temp[i] = ' ';break;
                                                                                }
                                                                                                }
                                                lclrblank(temp);
                                                No_Section = atoi(temp);
                                                 for (i = 0; i<250; i++){
                                                                temp[i] = fgetc(in);
                                                                if (temp[i] == '\n' ) {j = i;i = 250;}
                                                                                }
                                                                temp[j] = '\0';
                                                 for (i = 0; i < strlen(temp); i++ ) {
                                                                switch (temp[i]) {
                                                                                case '=': temp[i] = ' ';i = strlen(temp);
                                                break;
                                                                                default: temp[i] = ' ';break;
                                                                                }
                                                                                                }
                                                lclrblank(temp);
                                                AngularStep = atoi(temp);
                                                 for (i = 0; i<250; i++){
                                                                temp[i] = fgetc(in);
```

```
            if (temp[i] == '\n' ) {j = i;i = 250;}
                  }
            temp[j] = '\0';
 for (i = 0; i < strlen(temp); i++ ) {
            switch (temp[i]) {
                  case '=': temp[i] = ' ';i = strlen(temp);
break;
                  default: temp[i] = ' ';break;
                        }
                              }

 lclrblank(temp);
 Ra_Interval = atoi(temp);
 for (i = 0; i<250; i++){
            temp[i] = fgetc(in);
            if (temp[i] == '\n' ) {j = i;i = 250;}
                  }
            temp[j] = '\0';
 for (i = 0; i < strlen(temp); i++ ) {
            switch (temp[i]) {
                  case '=': temp[i] = ' ';i = strlen(temp);
break;
                  default: temp[i] = ' ';break;
                        }
                              }

 lclrblank(temp);
 Sec_Interval = atoi(temp);
 for (i = 0; i<250; i++){
            temp[i] = fgetc(in);
            if (temp[i] == '\n' ) {j = i;i = 250;}
                  }
            temp[j] = '\0';
 for (i = 0; i < strlen(temp); i++ ) {
            switch (temp[i]) {
                  case '=': temp[i] = ' ';i = strlen(temp);
break;
                  default: temp[i] = ' ';break;
                        }
                              }

 lclrblank(temp);
 PicMaxX = atoi(temp);
 for (i = 0; i<250; i++){
            temp[i] = fgetc(in);
            if (temp[i] == '\n' ) {j = i;i = 250;}
                  }
            temp[j] = '\0';
 for (i = 0; i < strlen(temp); i++ ) {
            switch (temp[i]) {
                  case '=': temp[i] = ' ';i = strlen(temp);
break;
                  default: temp[i] = ' ';break;
                        }
                              }

 lclrblank(temp);
 PicMaxY = atoi(temp);
 for (i = 0; i<250; i++){
            temp[i] = fgetc(in);
            if (temp[i] == '\n' ) {j = i;i = 250;}
                  }
            temp[j] = '\0';
 for (i = 0; i < strlen(temp); i++ ) {
            switch (temp[i]) {
                  case '=': temp[i] = ' ';i = strlen(temp);
break;
                  default: temp[i] = ' ';break;
                        }
                              }
 lclrblank(temp);
 PicLeft = atoi(temp);
 for (i = 0; i<250; i++){
            temp[i] = fgetc(in);
            if (temp[i] == '\n' ) {j = i;i = 250;}
                  }
            temp[j] = '\0';
 for (i = 0; i < strlen(temp); i++ ) {
            switch (temp[i]) {
                  case '=': temp[i] = ' ';i = strlen(temp);
break;
                  default: temp[i] = ' ';break;
                        }
                              }
 lclrblank(temp);
 PicTop = atoi(temp);
 for (i = 0; i<250; i++){
            temp[i] = fgetc(in);
            if (temp[i] == '\n' ) {j = i;i = 250;}
                  }
            temp[j] = '\0';
 for (i = 0; i < strlen(temp); i++ ) {
            switch (temp[i]) {
                  case '=': temp[i] = ' ';i = strlen(temp);
break;
                  default: temp[i] = ' ';break;
                        }
                              }
 lclrblank(temp);
 PicRight = atoi(temp);
 for (i = 0; i<250; i++){
            temp[i] = fgetc(in);
            if (temp[i] == '\n' ) {j = i;i = 250;}
                  }
            temp[j] = '\0';
 for (i = 0; i < strlen(temp); i++ ) {
            switch (temp[i]) {
                  case '=': temp[i] = ' ';i = strlen(temp);
break;
                  default: temp[i] = ' ';break;
                        }
                              }
 lclrblank(temp);
 PicBottom = atoi(temp);
 for (i = 0; i<250; i++){
            temp[i] = fgetc(in);
            if (temp[i] == '\n' ) {j = i;i = 250;}
                  }
            temp[j] = '\0';
 for (i = 0; i < strlen(temp); i++ ) {
            switch (temp[i]) {
                  case '=': temp[i] = ' ';i = strlen(temp);
break;
                  default: temp[i] = ' ';break;
                        }
```

```
                                                }
 lstrblank(temp);
 minAll = atof(temp);
  for (i = 0; i<250; i++){
                        temp[i] = fgetc(in);
                        if (temp[i] == '\n' ) {j = i;i = 250;}
                                }
                        temp[j] = '\0';
  for (i = 0; i < strlen(temp); i++ ) {
                        switch (temp[i]) {
                                case '=': temp[i] = ' ';i = strlen(temp);
break;
                                default: temp[i] = ' ';break;
                                        }
                                                }
 lstrblank(temp);
 maxAll = atof(temp);

 fscanf(in,"%s",temp);
 fscanf(in,"%s",temp);
 for(i=0;i<No_Profile;i++) fscanf(in,"%s",NProfile[i]);
 fscanf(in,"%s",temp);
 fscanf(in,"%s",temp);
 for(i=0;i<No_Section;i++) fscanf(in,"%s",NSection[i]);
            fclose(in);
PASS1:;

}


void DOT3(float x,float y, float z,int col)
{
float Rx,Ry,Rz;
    Tran3to2(x,y,z,&Rx,&Ry,&Rz);
    switch(PLAN){
                    case 0:
                                putpixel(Rx,Ry,col);
                                break;
                    case 1:
                                putpixel(Rx,Rz,col);
                                break;
                    case 2:
                                putpixel(Ry,Rz,col);
                                break;
                    default: break;
                    }

}


void Rplot(float x,float y, float z,int col)
{
float Rx,Ry,Rz;

    Tran3to2(x,y,z,&Rx,&Ry,&Rz);
    switch(PLAN){
                    case 0:
                                putpixel(Rx,Ry,col);
                                break;
                    case 1:
                                putpixel(Rx,Rz,col);
                                break;
                    case 2:
                                putpixel(Ry,Rz,col);
                                        break;
                                default: break;
                                }

}

void line_plot(float x,float y,float z,int indx,int col)
{
float Rx1,Rx2,Ry1,Ry2,Rz1,Rz2;
                Tran3to2(x,y,z,&Rx1,&Ry1,&Rz1);
                Tran3to2(x+indx,y,z,&Rx2,&Ry2,&Rz2);
                setcolor(col);
    switch(PLAN){
                                case 0:
                                        line(Rx1,Ry1,Rx2,Ry2);
                                        break;
                                case 1:
                                        line(Rx1,Rz1,Rx2,Rz2);
                                        break;
                                case 2:
                                        line(Ry1,Rz1,Ry2,Rz2);
                                        break;
                                default: break;
                                }
}


void line_plotR(float x,float y,float z,int indx,int col)
{
float Rx1,Rx2,Ry1,Ry2,Rz1,Rz2;
                Tran3to2(x,y,z,&Rx1,&Ry1,&Rz1);
                Tran3to2(x+indx,y,z,&Rx2,&Ry2,&Rz2);
                setcolor(col);
    switch(PLAN){
                                case 0:
                                        line(Rx1,Ry1,Rx2,Ry2);
                                        break;
                                case 1:
                                        line(Rx1,Rz1,Rx2,Rz2);
                                        break;
                                case 2:
                                        line(Ry1,Rz1,Ry2,Rz2);
                                        break;
                                default: break;
                                }
}

int Collor(float val)
{
int ret;
float r;
float def = 64;
   switch(ColMode){
                                case 0: case 64: def = 64; break;
                                case 1: case 128: def = 128; break;
                                case 2: case 192: def = 192; break;
                                default : break;
                                }
   r = (maxSel - val) / (maxSel - minSel) * def;
   if (r < 0) r = 0;
   if (r > def-4) r = def-4;
   ret = r + 64;
return ret;
}
```

```
float ValueCollor(int col)
{
float ret;
float r;
float def = 64;
    switch(ColMode){
                                        case 0: case 64: def = 64; break;
                                        case 1: case 128: def = 128; break;
                                        case 2: case 192: def = 192; break;
                                        default : break;
                                        }
     r =  minSel + (col * (maxSel - minSel)/def);
//    col = (maxSel - r)
//    if (r < 0) r = 0;
//    if (r > def-4) r = def-4;
   ret = r;
return ret;
}

void showdisk(int p)
{
FILE *in;
int i,ii,j,jj,k,col;
float val;
char temp[255];
          if ((in = fopen(NSection[p], "rt"))  == NULL)
          {
            printf("Cannot open input file.\n");
            exit(0);
          }
          i = p;
                    jj=0;
                    for(ii=0;ii<400;ii++)
                    {
                              temp[ii] = fgetc(in);
                              if (temp[ii] == '\n') jj++;
                              if (jj == 5) ii = 400;
                    }

          for(j=0;j<No_Raysum;j++)
          {
            for(k=0;k<No_Raysum;k++)
            {
                    fscanf(in,"%f",&val);
                    if(val != -1000)
                    {
                      if( (val > minSel )&&(val < maxSel) ){
                              col = Collor(val);
                              DOT3(k,j,i*Sec_Interval,col);
                              }
                    }
            }
          }
          fclose(in);
}



int Ax=0,Ay=0,Az=0;
int SII = 0;
int ADI = 0;
int AAV = 0;
int ADA = 0;


void Rshowdisk(int p)
{
FILE *in;
int i,ii,j,jj,k,col;
float val;
float Rx,Ry,Rz;
float Ang;
float x,y,z;
char temp[255];

          if ((in = fopen(NSection[p], "rt"))  == NULL)
          {
            printf("Cannot open input file.\n");
            exit(0);
          }
          i = p;
                    jj=0;
                    for(ii=0;ii<400;ii++)
                    {
                              temp[ii] = fgetc(in);
                              if (temp[ii] == '\n') jj++;
                              if (jj == 5) ii = 400;
                    }
z = p * Sec_Interval;
//z = p;
          for(j=0;j<No_Raysum;j++)
          {  z = p*Sec_Interval; y = j;
            for(k=0;k<No_Raysum;k++)
            {
                    fscanf(in,"%f",&val);
                    if(val != -1000)
                    {
                      if( (val > minSel )&&(val < maxSel) ){
                              col = Collor(val);
                              Ang = Ax * 6.283185 * 2.7777778E-
3;
                              CosRotAngle = cos(Ang);
SinRotAngle = sin(Ang);
                              x = k; y = j; z = p*Sec_Interval;
                              RotateX(x-128,y-128,z-
30,&Rx,&Ry,&Rz);
                              Ang = Ay * 6.283185 * 2.7777778E-
3;
                              CosRotAngle = cos(Ang);
                              SinRotAngle = sin(Ang);
                              x = Rx; y = Ry; z = Rz;
                              RotateY(x,y,z,&Rx,&Ry,&Rz);
                              Ang = Az * 6.283185 * 2.7777778E-
3;
                              CosRotAngle = cos(Ang);
                              SinRotAngle = sin(Ang);
                              x = Rx; y = Ry; z = Rz;
                              RotateZ(x,y,z,&Rx,&Ry,&Rz);
                              x = Rx; y = Ry; z = Rz;
                              DOT3(x+128,y+128,z+30,col);
                              }
                    }
            }
          }
          fclose(in);
}
```

```
void Rshowpins(int p)
{
FILE *in;
int i,ii,j,jj,k,col;
float val;
float Rx,Ry,Rz;
float Ang;
float x,y,z;
char temp[255];

          if ((in = fopen(NSection[p], "rt"))  == NULL)
          {
            printf("Cannot open input file.\n");
            exit(0);
          }
          i = p;
                    jj=0;
                    for(ii=0;ii<400;ii++)
                    {
                              temp[ii] = fgetc(in);
                              if (temp[ii] == '\n') jj++;
                              if (jj == 5) ii = 400;
                    }
z = p * Sec_Interval;
//z = p;
          for(j=0;j<No_Raysum;j++)
          {   z = p*Sec_Interval; y = j;
              for(k=0;k<No_Raysum;k++)
              {
                    fscanf(in,"%f",&val);
                    if(val != -1000)
                    {
                     if((k > No_Raysum/2) || (j > No_Raysum/2))
                      if( (val > minSel )&&(val < maxSel) ){
                              col = Collor(val);
                              Ang = Ax * 6.283185 * 2.777777E-
3;
                              CosRotAngle = cos(Ang);
SinRotAngle = sin(Ang);
                              x = k; y = j; z = p*Sec_Interval;
                              RotateX(x-128,y-128,z-
30,&Rx,&Ry,&Rz);
                              Ang = Ay * 6.283185 * 2.777777E-
3;
                              CosRotAngle = cos(Ang);
                              SinRotAngle = sin(Ang);
                              x = Rx; y = Ry; z = Rz;
                              RotateY(x,y,z,&Rx,&Ry,&Rz);
                              Ang = Az * 6.283185 * 2.777777E-
3;
                              CosRotAngle = cos(Ang);
                              SinRotAngle = sin(Ang);
                              x = Rx; y = Ry; z = Rz;
                              RotateZ(x,y,z,&Rx,&Ry,&Rz);
                              x = Rx; y = Ry; z = Rz;
//                            if((x>No_Raysum/2) ||
(y>No_Raysum/2))
                               DOT3(x+128,y+128,z+30,col);
                             }
                    }
              }
          }
          fclose(in);
```

```
}

void showObject(void)
{
int i,j;
float x=1,y=1,z=1;
float Rx,Ry,Rz,Ang;
char temp[255];
          switch(ColMode){
                                        case 0:
                                        case 64:
                                          if(RealOb) BW64(1); else
BW64R(1);
                                          for(i=0;i<64;i++)
                                          {
                                                    setcolor(64+i);
          line(180+i*2,10,180+i*2,18);
          line(181+i*2,10,181+i*2,18);
                                          }
          setrgbpalette(1,63,63,63);setcolor(1);
          line(179,9,179,19);line(179,9,308,9);
          line(308,9,308,19);line(179,19,308,19);
                                          break;
                                        case 1:
                                        case 128:
                                          if(RealOb) BW128(); else
BW128R();
                                          for(i=0;i<128;i++)
                                          {
                                                    setcolor(64+i);
          line(180+i,10,180+i,18);
                                          }
          setrgbpalette(1,63,63,63);setcolor(1);
          line(179,9,179,19);line(179,9,308,9);
          line(308,9,308,19);line(179,19,308,19);
                                          break;
                                        case 2:
                                        case 192:
                                          if(RealOb) BW192(); else
BW192R();
                                          for(i=0;i<192;i++)
                                          {
                                                    setcolor(64+i);
          line(180+i,10,180+i,18);
                                          }
          setrgbpalette(1,63,63,63);setcolor(1);
                                                    setcolor(1);
          line(179,9,179,19);line(179,9,372,9);
          line(372,9,372,19);line(179,19,372,19);
                                          break;
                                        default:   break;
                                        }
                                        Ang = Ax * 6.283185 * 2.7777778E-
3;
                                        CosRotAngle = cos(Ang);
SinRotAngle = sin(Ang);
                                        RotateX(x-128,y-128,z-
30,&Rx,&Ry,&Rz);
                                        Ang = Ay * 6.283185 * 2.7777778E-
3;
                                        CosRotAngle = cos(Ang);
                                        SinRotAngle = sin(Ang);
                                        x = Rx; y = Ry; z = Rz;
                                        RotateY(x,y,z,&Rx,&Ry,&Rz);
                                        Ang = Az * 6.283185 * 2.7777778E-
3;
                                        CosRotAngle = cos(Ang);
                                        SinRotAngle = sin(Ang);
                                        x = Rx; y = Ry; z = Rz;
                                        RotateZ(x,y,z,&Rx,&Ry,&Rz);
 if(Rz >= 0)
 {
              for(i=0;i<No_Section;i+=1)
              {
                        setcolor(61);
//                      sprintf(temp,"Section = %d",i+1);
                        sprintf(temp,"                    ");
                        outtextxy(410,150,temp);
                        setcolor(31);
                        sprintf(temp,"Section = %d",i);
                        outtextxy(410,150,temp);
               Rshowdisk(i);
              }
 }
 else
              for(i=No_Section-1;i>=0;i-=1)
              {
                        setcolor(61);
//                      sprintf(temp," Section = %d",i+1);
                        sprintf(temp,"                    ");
                        outtextxy(410,150,temp);
                        setcolor(31);
                        sprintf(temp,"Section = %d",i);
                        outtextxy(410,150,temp);
               Rshowdisk(i);
              }

}

void show_pins(void)
{
int i,j;
float x=1,y=1,z=1;
float Rx,Ry,Rz,Ang;
char temp[255];
                                        PALETTE();
                                        Ang = Ax * 6.283185 * 2.7777778E-
3;
                                        CosRotAngle = cos(Ang);
SinRotAngle = sin(Ang);
```

```
                                        RotateX(x-128,y-128,z-
30,&Rx,&Ry,&Rz);
                                        Ang = Ay * 6.283185 * 2.7777778E-
3;
                                        CosRotAngle = cos(Ang);
                                        SinRotAngle = sin(Ang);
                                        x = Rx; y = Ry; z = Rz;
                                        RotateY(x,y,z,&Rx,&Ry,&Rz);
                                        Ang = Az * 6.283185 * 2.7777778E-
3;
                                        CosRotAngle = cos(Ang);
                                        SinRotAngle = sin(Ang);
                                        x = Rx; y = Ry; z = Rz;
                                        RotateZ(x,y,z,&Rx,&Ry,&Rz);
 if(Rz >= 0)
 {
              for(i=0;i<No_Section;i+=1)
              {
                        setcolor(61);
//                      sprintf(temp,"Section = %d",i-1);
                        sprintf(temp,"                    ");
                        outtextxy(410,150,temp);
                        setcolor(31);
                        sprintf(temp,"Section = %d",i);
                        outtextxy(410,150,temp);
              Rshowpins(i);
              }
 }
 else
              for(i=No_Section-1;i>=0;i-=1)
              {
                        setcolor(61);
                        sprintf(temp,"                    ");
//                      sprintf(temp,"Section = %d",i+1);
                        outtextxy(410,150,temp);
                        setcolor(31);
                        sprintf(temp,"Section = %d",i);
                        outtextxy(410,150,temp);
               Rshowpins(i);
              }
}

void PALETTE(void)
{
int i;
              switch(ColMode){
                                        case 0:
                                        case 64:
                                          if(RealOb) BW64(1); else
BW64R(1);
                                          for(i=0;i<64;i++)
                                          {
                                                    setcolor(64+i);
              line(180+i*2,10,180+i*2,18);
              line(181+i*2,10,181+i*2,18);
                                          }
              setrgbpalette(1,63,63,63);setcolor(1);
              line(179,9,179,19);line(179,9,308,9);
```

```
            line(308,9,308,19);line(179,19,308,19);
                                    break;
                                case 1:
                                case 128:
                                    if(RealOb) BW128(); else
BW128R();
                                    for(i=0;i<128;i++)
                                    {
                                            setcolor(64+i);
            line(180+i,10,180+i,18);
                                    }
            setrgbpalette(1,63,63,63);setcolor(1);
            line(179,9,179,19);line(179,9,308,9);

            line(308,9,308,19);line(179,19,308,19);
                                    break;
                                case 2:
                                case 192:
                                    if(RealOb) BW192(); else
BW192R();
                                    for(i=0;i<192;i++)
                                    {
                                            setcolor(64+i);
            line(180+i,10,180+i,18);
                                    }
            setrgbpalette(1,63,63,63);setcolor(1);
                                            setcolor(1);
            line(179,9,179,19);line(179,9,372,9);

            line(372,9,372,19);line(179,19,372,19);
                                    break;
                                default:    break;
                        }
}

void ClsBox(void)
{
int i;
            setcolor(0);
            for(i=0;i<192;i++) line(180+i,22,180+i,30);
}

void gr3d(void)
{
int gdriver, gmode, errorcode;
int maxx,maxy;
int bo,p;
int i,j,k,t,col;
int ii,jj,o,q=1;
char st[50];
char temp[255];
float val,valS,valF;
int indx;
float x,y,z;
float x1,x2,y1,y2,z1,z2;
```

```
FILE *in;

gr3dI();
gdriver = installuserdriver("SVGA256", Detect);
gmode = ScrMode;
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult();

if (errorcode != grOk)  /* an error occurred */
{
  printf("Graphics error: %s\n", grapherrormsg(errorcode));
  printf("Press any key to halt:");
  getch();
  exit(1);           /* return with error code */
}

                setrgbpalette(1,63,63,63);setcolor(1);
                maxx = getmaxx() -1; maxy = getmaxy() -1;
                block(0,0,maxx,maxy);
                outtextxy(5,10,"Display all color ->");
                outtextxy(5,22,"    Select color ->");
                line(179,21,179,31);line(179,21,372,21);
                line(372,21,372,31);line(179,31,372,31);
                setrgbpalette(51,16,16,0);
                setcolor(51);
                for(i=0;i<192;i++) line(180+i,22,180+i,30);

                ColMode = 192;
                PALETTE();
//box();
    minSel = minAll; maxSel = maxAll;
setcolor(15);
//outtextxy(5,10,"Display all color ->");
for(i=31;i<50;i++)
{
   setrgbpalette(i,63,63,63);
}
setrgbpalette(61,0,0,0);
p = 0;

setcolor(31);
                        sprintf(temp,"Rotate Axist X = %i",Ax);
                        outtextxy(410,100,temp);
                        sprintf(temp,"Rotate Axist Y = %i",Ay);
                        outtextxy(410,110,temp);
                        sprintf(temp,"Rotate Axist Z = %i",Az);
                        outtextxy(410,120,temp);
                        SII = ColMode/2;
                        sprintf(temp,"Interest Interval = %i",SII);
                        outtextxy(410,130,temp);
                        ADI = ColMode/2;
                        sprintf(temp,"Deviation Interval = %i",ADI);
                        outtextxy(410,140,temp);
//                      sprintf(temp,"Air Value = %i",AAV);
//                      sprintf(temp,"Air Value = %i",minAll);
//                      outtextxy(410,150,temp);
//                      ADA = 1;
//                      sprintf(temp,"Air Deviation = %i",ADA);
//                      sprintf(temp,"Air Deviation = %i",maxAll);
//                      outtextxy(410,160,temp);
```

```
setcolor(4);
outtextxy(350,360,"KEY CONTROL");
setcolor(30);
outtextxy(350,370,"<X> is Rotate Axis X");
outtextxy(350,380,"<Y> is Rotate Axis Y");
outtextxy(350,390,"<Z> is Rotate Axis Z");
outtextxy(350,400,"<S> is Select Interest Interval");
outtextxy(350,410,"<D> is Assign Deviation Interval");
outtextxy(350,420,"<I> is Invert Color");
//outtextxy(350,420,"<E> is Assign Deviation of Air");
outtextxy(350,430,"<J> is Display Model Two");
outtextxy(350,440,"<G> is Calculate");
outtextxy(350,450,"<P> is Display Section");
outtextxy(350,460,"<R> Back to Display Menu");


p = 1;
do{
  c = getch(); if(!c) c=getch();
  switch(c){
                case 'i':
                case 'I':
                                RealOb = !RealOb;
                                PALETTE();
                                break;
                case 'j':
//                              minSel = minAll; maxSel = maxAll;
//                              SII = ColMode/2;
//                              ADI = ColMode/2;
                                if((SII-ADI) < 0 ) bc = 0; else bc = SII - ADI;
                                minSel = ValueCollor(bc);
                                if((SII+ADI) > ColMode) bc = ColMode;else bc =
SII+ADI;

                                maxSel = ValueCollor(bc);
                                setcolor(61);
                                for(i=0;i<400;i++)
                                          line(5+i,32,5+i,340);
                                for(i=0;i<100;i++)
                                          line(5,340+i,340,340+i);
                                show_pine();
                                break;
                case 'G':
                                 setcolor(61);
                                 sprintf(temp,"                    ");
                                        outtextxy(410,100,temp);
                                        outtextxy(410,110,temp);
                                        outtextxy(410,120,temp);
                                        outtextxy(410,130,temp);
                                        outtextxy(410,140,temp);
                                        outtextxy(410,150,temp);
                                setcolor(31);
                                minSel = minAll; maxSel = maxAll;
//                              Ax = 0; Ay = 0; Az =0;
                                        sprintf(temp,"Rotate Axist X =
%i",Ax);

                                        outtextxy(410,100,temp);
                                        sprintf(temp,"Rotate Axist Y =
%i",Ay);

                                        outtextxy(410,110,temp);
                                        sprintf(temp,"Rotate Axist Z =
%i",Az);

                                        outtextxy(410,120,temp);
                                        SII = ColMode/2;
                                        sprintf(temp,"Interest Interval =
%i",SII);

                                        outtextxy(410,130,temp);
                                        ADI = ColMode/2;
                                        sprintf(temp,"Deviation Interval =
%i",ADI);

                                        outtextxy(410,140,temp);
                                        setcolor(51);
                                        for(i=SII-ADI;i<SII+ADI;i++)
line(180+i,22,180+i,30);
                case 'g':
                                  if((SII-ADI) < 0 ) bc = 0; else bc = SII - ADI;
                                  minSel = ValueCollor(bc);
                                  if((SII+ADI) > ColMode) bc = ColMode;else bc =
SII+ADI;

                                  maxSel = ValueCollor(bc);
                                  setcolor(61);
                                  for(i=0;i<400;i++)
                                            line(5+i,32,5+i,340);
                                  for(i=0;i<100;i++)
                                            line(5,340+i,340,340+i);
                                  showObject();
                                  SII = ColMode/2;
                                  ADI = ColMode/2;
                                ClsBox();
                                setcolor(51);
                                for(i=SII-ADI;i<SII+ADI;i++)
line(180+i,22,180+i,30);

                                  break;
                case 'p':
                                  if(No_Section > 10) p+=5;else p+= 1;
                                  if(p>= No_Section) p = No_Section -1;
                                   setcolor(61);
                                   sprintf(temp,"                ");
                                   outtextxy(410,150,temp);
                                   setcolor(31);
                                   sprintf(temp,"Section = %i",p);
                                   outtextxy(410,150,temp);
                                  Rshowdisk(p);
                                break;
                case 'P':
                                  if(No_Section > 10) p-=4;else p -=2;
                                  if(p < 0) p = 0;
                                  setcolor(61);
                                  sprintf(temp,"                ");
                                  outtextxy(410,150,temp);
                                  setcolor(31);
                                  sprintf(temp,"Section = %i",p);
                                  outtextxy(410,150,temp);
                                  Rshowdisk(p); break;
                case 'e':
                                setcolor(61);
                                sprintf(temp,"Air Deviation = %i",ADA);
                                outtextxy(410,160,temp);
                                ADA += 1; if (ADA > 10) ADA = 10;
                                setcolor(31);
                                sprintf(temp,"Air Deviation = %i",ADA);
                                outtextxy(410,160,temp);
                                break;
                case 'E':
                                setcolor(61);
                                sprintf(temp,"Air Deviation = %i",ADA);
                                outtextxy(410,160,temp);
                                ADA -= 1; if (ADA < 1) ADA = 1;
                setcolor(31);
                sprintf(temp,"Air Deviation = %i",ADA);
                outtextxy(410,160,temp);
                break;
        case 'a':
                setcolor(61);
                sprintf(temp,"Air Value = %i",AAV);
outtextxy(410,150,temp);
                setrgbpalette(AAV+64,AAV/3,AAV/3,AAV/3);
                AAV += 1; if (AAV >= ColMode-1) AAV =
ColMode-1;
                setrgbpalette(AAV+64,0,63,63);
                setcolor(31);
                sprintf(temp,"Air Value = %i",AAV);
outtextxy(410,150,temp);
                break;
        case 'A':
                setcolor(61);
                sprintf(temp,"Air Value = %i",AAV);
outtextxy(410,150,temp);
                setrgbpalette(AAV+64,AAV/3,AAV/3,AAV/3);
                AAV -= 1; if (AAV < 0) AAV = 0;
                setrgbpalette(AAV+64,0,63,63);
                setcolor(31);
                sprintf(temp,"Air Value = %i",AAV);
outtextxy(410,150,temp);
                break;
        case 's':
                setcolor(61);
                sprintf(temp,"Interrest Interval = %i",SII);
outtextxy(410,130,temp);
                SII += 1; if (SII > 191) SII = 191;
                if((SII + ADI) > 192) SII -= 1 ;
                if((SII - ADI) < 0 ) SII -= 1 ;
                ClsBox();
                setcolor(51);
                for(i=SII-ADI;i<SII+ADI;i++)
line(180+i,22,180+i,30);
                setcolor(31);
                sprintf(temp,"Interrest Interval = %i",SII);
outtextxy(410,130,temp);
                break;
        case 'S':
                setcolor(61);
                sprintf(temp,"Interrest Interval = %i",SII);
outtextxy(410,130,temp);
                SII -= 1; if (SII < 1) SII = 1;
                if((SII + ADI) > 192) SII += 1 ;
                if((SII - ADI) < 1 ) SII += 1 ;
                ClsBox();
                setcolor(51);
                for(i=SII-ADI;i<SII+ADI;i++)
line(180+i,22,180+i,30);
                setcolor(31);
                sprintf(temp,"Interrest Interval = %i",SII);
outtextxy(410,130,temp);
                break;
        case 'x':
                setcolor(61);
                sprintf(temp,"Rotate Axist X = %i",Ax);
outtextxy(410,100,temp);
                setcolor(31);
                Ax += 10; if (Ax > 359 ) Ax =0;
                sprintf(temp,"Rotate Axist X = %i",Ax);
outtextxy(410,100,temp);
                break;
        case 'X':
                setcolor(61);
                sprintf(temp,"Rotate Axist X = %i",Ax);
outtextxy(410,100,temp);
                setcolor(31);
                  Ax -= 10; if (Ax < 1 ) Ax =360;
                sprintf(temp,"Rotate Axist X = %i",Ax);
outtextxy(410,100,temp);
                break;
        case 'y':
                setcolor(61);
                sprintf(temp,"Rotate Axist Y = %i",Ay);
outtextxy(410,110,temp);
                setcolor(31);
                Ay += 10; if (Ay > 359 ) Ay =0;
                sprintf(temp,"Rotate Axist Y = %i",Ay);
outtextxy(410,110,temp);
                break;
        case 'Y':
                setcolor(61);
                sprintf(temp,"Rotate Axist Y = %i",Ay);
outtextxy(410,110,temp);
                setcolor(31);
                Ay -= 10; if (Ay < 1 ) Ay = 360;
                sprintf(temp,"Rotate Axist Y = %i",Ay);
outtextxy(410,110,temp);
                break;
        case 'z':
                setcolor(61);
                sprintf(temp,"Rotate Axist Z = %i",Az);
outtextxy(410,120,temp);
                setcolor(31);
                Az += 10; if (Az > 359 ) Az =0;
                sprintf(temp,"Rotate Axist Z = %i",Az);
outtextxy(410,120,temp);
                break;
        case 'Z':
                setcolor(61);
                sprintf(temp,"Rotate Axist Z = %i",Az);
outtextxy(410,120,temp);
                setcolor(31);
                Az -= 10; if (Az < 1 ) Az = 360;
                sprintf(temp,"Rotate Axist Z = %i",Az);
outtextxy(410,120,temp);
                break;
        case 'd':
                setcolor(61);
                sprintf(temp,"Deviation Interval = %i",ADI);
outtextxy(410,140,temp);
                ADI += 1; if(ADI > 96) ADI = 96;
                ClsBox();
                setcolor(51);
                for(i=SII-ADI;i<SII+ADI;i++)
line(180+i,22,180+i,30);
                setcolor(31);
                sprintf(temp,"Deviation Interval = %i",ADI);
outtextxy(410,140,temp);
                  break;
        case 'D':
                setcolor(61);
```

```
                        sprintf(temp,"Deviation Interval = %i",ADI);
outtextxy(410,140,temp);
                        ADI -= 1; if(ADI < 1) ADI = 1;
                        ClsBox();
                        setcolor(51);
                        for(i=SII-ADI;i<SII+ADI;i++)
line(180+i,22,180+i,30);
                        setcolor(31);
                        sprintf(temp,"Deviation Interval = %i",ADI);
outtextxy(410,140,temp);
                         break;
              case 'R': case 'r': q = !q; break;
              default: break;
              }
  }while(q);
closegraph();
}


void showall(void)
{
int gdriver,gmode,errorcode;
FILE *in;
int p,i,j,ii,jj,pp;
int size = 64;
int Nopic = 12;
int cc,dd,col;
float val;
char temp[255];
int key;

gr3d();
gdriver = installuserdriver("SVGA256", Detect);
gmode = SxMode;
initgraph(&gdriver, &gmode, "");

/* read result of initialization */
errorcode = graphresult();

if (errorcode != grOk)  /* an error occurred */
{
  printf("Graphics error: %s\n", grapherrormsg(errorcode));
  printf("Press any key to halt:");
  getch();
  exit(1);           /* return with error code */
}
//PALETTE();
 BW192();
 minSel = minAll;
 maxSel = maxAll;
for(p=0;p<No_Section;p++)
{
          if ((in = fopen(NSection[p], "rt"))  == NULL)
          {
            printf("Cannot open input file.\n");
          }
                  jj=0;
                  for(ii=0;ii<1000;ii++)
                  {
                            temp[ii] = fgetc(in);
                            if (temp[ii] == '\n') jj++;
                            if (jj == 5) ii = 1000;
                  }
          cc = p % 12;
```

```
          cc /= 4;
          dd = p % 4;
            for(i=0;i<No_Raysum;i++)
                      for(j=0;j<No_Raysum;j++)
                      {
                        fscanf(in,"%f",&val);
                        if(val != -1000)
                        {
                                  col = Collor(val);
                                  if((i%2) == 0)
                                   if((j%2) == 0)
putpixel(i/2+(160*dd),j/2+(150*cc),col);
                        }
                      }
          fclose(in);
          sprintf(temp,"Section %i",p+1);
          outtextxy(30+(160*dd),135+(150*cc),temp);
          switch(p){
                      case 11:  case 23:
                      case 35:  case 47:
                      case 59:
                                outtextxy(230,470,"Press any key to
continue");
                                key = getch();
                                switch(key){
                                          case 'q':
                                          case 'Q': p =
No_Section;
                                          case 'n':
                                          case 'N': break;
                                          case 'b':
                                          case 'B':
          if(p==23) p = -1;

          if(p==35) p = 11;

          if(p==47) p = 23;

          if(p==59) p = 35;

          break;
                                          default: break;
                                          };
                               cleardevice();
                               break;
                     default:
                               break;
                     }
       if(p == No_Section-1){
                               outtextxy(230,470,"Press any key to
continue");
                               key = getch();
                               switch(key){
                                          case 'q':
                                          case 'Q': p =
No_Section;
                                          case 'n':
                                          case 'N': break;
                                          case 'b':
                                          case 'B':
          if(p>59) p = 47;
if(p>47) p = 35;

if(p>35) p = 23;

if(p>23) p = 11;

cleardevice();
```

```
                    break;
                                                  default: break;
                                                  };
                              }
}
//scr2pcx("\\showall");
closegraph();
}
```

ประวัติผู้เขียน

นาย เชิงชาย สร้อยเพ็ชร เกิดวันที่ 22 กุมภาพันธ์ พ.ศ. 2513 ที่อำเภอ เมือง จังหวัด จันทบุรี สำเร็จการศึกษาปริญญาตรีวิทยาศาสตรบัณฑิต ภาควิชา คณิตศาสตร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2536 สำเร็จการศึกษาประกาศนียบัตรบัณฑิต สาขาวิชา นิวเคลียร์เทคโนโลยี ภาควิชา นิวเคลียร์เทคโนโลยี คณะ วิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2537 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตร์มหาบัณฑิต สาขา นิวเคลียร์เทคโนโลยี ภาควิชา นิวเคลียร์เทคโนโลยี คณะ วิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2538