TERM PROJECT

SKYLINE

컴퓨터 공학과 964524 김진홍
email: s964524@ce.cecc.hallym.ac.kr

아래 프로그램은 여러개의 사각형이 화면에 wireframe으로 그려질때, 그것들의 외곽선(skyline)만을 뽑아내는 프로그램이다. 본 프로젝트는 난이도를 약간 낮추어 사각형의 밑바닥을 땅에 붙이도록 하였으나, 내가 생각하기에는 실제로 VLSI에 이용되는 공중에 떠 있는 경우를 포함한 알고리즘과 땅에 모두 붙어있는 경우의 알고리즘이 별로 다를것 같지 않아, 나는 처음부터 공중에 떠 있는 경우를 포함에 제작에 임하였다.

아래 프로그램은 Text로 이루어진 데이타 파일을 필요로 한다. 위에서 언급했지만, 난 공중에 떠 있는 경우도 고려 하였기 때문에 사각형의 4개 좌표가 모두 필요하다. 기본적으로 5개의 예제를 마련해 놓았다.

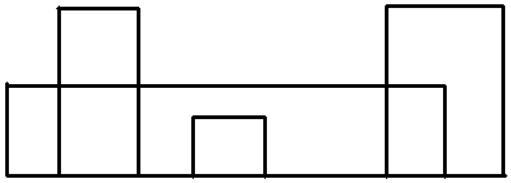


Figure 1: 기본 skyline 데이타 (1번예제)

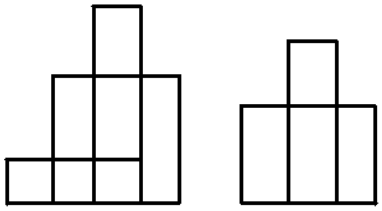


Figure 2: 두개의 독립된 skyline이 존재하는 경우(2번예제)

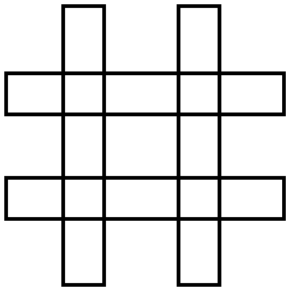
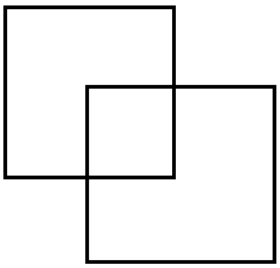


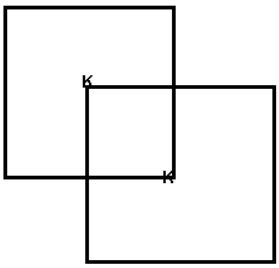
Figure 3: 공중에 떠 있고, 4개의 물체가 서로 둘러싸 내부에 영역이 생긴경우 (5번예제)

알고리즘 :

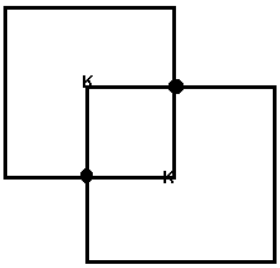
다음 예제를 통해 알고리즘을 설명하겠다.



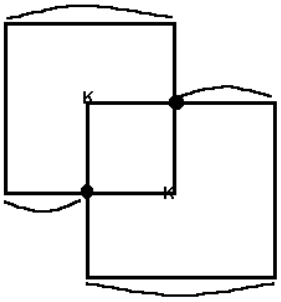
* 두개의 사각형



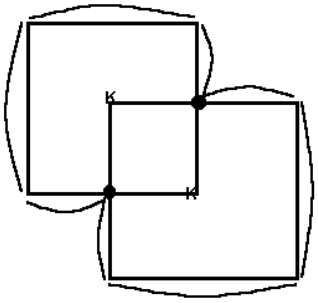
1. 각 사각형마다 자신의 영역안에 들어있는 모든 점을 kill한다. kill되면 그 점은 사용되지 않는다.



2. 각 사각형끼리의 교차점을 찾아낸다.



3. x좌표가 가장 왼쪽인 것부터 오른쪽으로 scan하면서 가장 가까이 위치한 점과 연결한다. 한번 연결된것은 다시 연결하지 않는다.

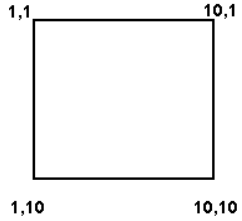


4. y좌표가 가장 위인것부터 아래쪽으로 scan하며 가장 가까이 위치한 점과 연결한다. 역시 한번 연결된것은 다시 연결하지 않는다.

기본 알고리즘은 위의 4단계이다. 위 4단계를 거치고 나면 모든 점들이 서로 연결이 되는데, 그중 시작점부터 자신과 연결된점들을 추적해 나가면 된다. 크게 나누어 이렇게 4단계로 나뉘지만, 완벽히 수행하기 위해서는 좀더 복잡한 과정이 필요하다.

사각형의 좌표 데이타 입력시 주의점 :

이 프로그램은 프로그램 작성의 편의를 위해 SCREEN좌표계를 사용하였다. SCREEN좌표계에서는 X축은 오른쪽으로 갈수록 커지나(다른 좌표계와 동일), Y축은 지게 되어있다. 데이타는 하나의 사각형당 xy좌표가 4개가 필요하므로, 한줄에 8개의 숫자가 들어가도록 되어있다. 입력 순서는 LeftBottom, LeftTop, RightTop, RightBottom 이다.



즉, 이와같은 사각형의 데이타는...

1 10 1 1 10 1 10 10 으로 입력한다.
다음 사각형은 이와 같은 방식으로 다음줄에 넣으면 된다.

* 여기서는 교수님의 권고대로 Maximum 500개의 점(125개의 사각형)을 다룬다.
* 아래 프로그램은 데이타와 출력데이타를 그래픽으로도 보여줄 수 있도록 되어 있으므로 약간 길이가 긴 편이다. 단, 여기서 사용하는 그래픽 해상도는 320x200이므로, 크기가 320x200을 초과하는 사각형은 제대로 표시될 수 없다.
* 만일 입력 데이타를 그래픽으로 볼때, 제대로 그려지지 않는다면 그것은 입력된 값이 잘못되었을 경우가 대부분이다. 만일 그 상태로 실행시킨다면, 프로그램이 다음 점을 추적하는데에 실패해 무한루프를 돌 수도 있다.
* 다시한번, 여기서 사용되는 좌표계는 SCREEN좌표계라는 것을 알린다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mem.h>
#include <conio.h>

#define putdot(x,y,c) *((char far *)0xa0000000 + (unsigned)((x) + (y)*320))=c

typedef struct DOTINFO      /* 점을 나타내는 구조체 */
{
  int x,y;
  int killed;
  int vlink, hlink;
} DOTINFO;

#define MAXDOT          4*125         /* MAX: 500 dots (125 rectangles) */

DOTINFO dot[MAXDOT];        /* 입력 데이타를 저장할 곳 */
DOTINFO cdot[MAXDOT];       /* 각 사각형들을 이루는 라인의 교차점을 저장할 곳 */
DOTINFO ws[MAXDOT*2];       /* 입력하여 처리한 점들과 교차점을 합하여 저장할 곳 */
DOTINFO bws[MAXDOT*2];      /* 임시 저장 장소 */
int visit[MAXDOT*2];        /* 각 점을 추적시 '방문' 표시를 해둔다. 이것은 독립된
                               사각형들이 나올경우 추적되지 않는곳을 찾아내기 위해
                               사용된다 */
```

```
int dotidx=0;
int crsidx=0;
int wsidx=0;

void killhiddendot(void);           /* 각 상자안에 들어오는 점들을 kill한다 */
void killhiddencdot(void);          /* 각 상자안에 들어오는 교차점들을 kill 한다 */
void killsamecdot(void);            /* 가끔 중복되어 출력되는 좌표를 제거한다 */
void killsamewsdot(void);           /* 위와 같은 기능을 ws배열에 대해 처리한다 */
void findcdot(void);                /* 교차점을 찾아냄 */
void makews(void);                  /* ws배열을 준비시켜 줌 */
void prepare(void);                 /* 각점을 연결하기전 마지막 준비 */
void quicksort(int, int, int);
void swapdata(DOTINFO *, DOTINFO *);
void link_horizontal(void);         /* 각 점을 수평으로 연결한다 */
void link_vertical(void);           /*  ...    수직으로  ...  */
int findRnearest(int);              /* 수평으로 가장 가까운 점을 찾음 */
int findBnearest(int);              /* 수직으로 ... */
int bws2ws(int);                    /* bws배열에서 사용되는 위치좌표를 ws에 맞도록 변환 */
int notvisit(void);                 /* 추적되지 않는곳을 찾아냄 */
void showlist(void);                /* 결과 */
void readdata(char *);              /* 데이타를 파일로부터 읽음 */

void drawdatagrp(void);             /* 읽은 데이타를 그래픽으로 표현 */
void drawoutgrp(void);              /* 결과 데이타를 ... */
void drawhline(int, int, int, int); /* 수평선을 긋는다 */
void drawvline(int, int, int, int); /* 수직선을 ... */

void main(void)
{
  char name[60];
  char c;

  memset(dot, 0, sizeof(DOTINFO)*MAXDOT);
  memset(cdot, 0, sizeof(DOTINFO)*MAXDOT);
  memset(visit, 0, sizeof(int)*MAXDOT*2);

  printf("Input data file name: ");
  scanf("%s", &name);

  readdata(name);

  printf("Do you want to see the data in graphic? (y/n)\n");
  c=getch();
  if (c == 'y' || c == 'Y') drawdatagrp();

  killhiddendot();
  findcdot();
  killhiddencdot();
  killsamecdot();
  makews();
  killsamewsdot();
  prepare();
```

```
  link_horizontal();
  link_vertical();

  showlist();

  printf("Do you want to see the output in graphic? (y/n)\n");
  c=getch();
  if (c == 'y' || c == 'Y') drawoutgrp();
}

void killhiddendot()
{
  int t, i;

  for (t=0; t<dotidx; t+=4)
    for (i=0; i<dotidx; i++)
    {
        if ((dot[t+0].y >= dot[i].y) &&
           (dot[t+1].y <= dot[i].y) &&
           (dot[t+1].x <= dot[i].x) &&
           (dot[t+2].x >= dot[i].x))
           if (((dot[t+0].x == dot[i].x) && (dot[t+0].y == dot[i].y)) ||
               ((dot[t+1].x == dot[i].x) && (dot[t+1].y == dot[i].y)) ||
               ((dot[t+2].x == dot[i].x) && (dot[t+2].y == dot[i].y)) ||
               ((dot[t+3].x == dot[i].x) && (dot[t+3].y == dot[i].y)));
           else
               dot[i].killed=1;
    }
}

void killhiddencdot()
{
  int t, i;

  for (t=0; t<dotidx; t+=4)
    for (i=0; i<crsidx; i++)
        if ((dot[t+0].y > cdot[i].y) &&
           (dot[t+1].y < cdot[i].y) &&
           (dot[t+1].x < cdot[i].x) &&
           (dot[t+3].x > cdot[i].x))
           cdot[i].killed=1;
}

void killsamecdot()
{
  int t, i;

  for (t=0; t<crsidx; t++)
    for (i=t+1; i<crsidx; i++)
    if (t != i)
      if ((cdot[t].x == cdot[i].x) &&
          (cdot[t].y == cdot[i].y))
         cdot[i].killed=1;
```

```
}

void killsamewsdot()
{
  int t, i;

  for (t=0; t<wsidx; t++)
    for (i=t+1; i<wsidx; i++)
    if (t != i)
      if ((ws[t].x == ws[i].x) &&
          (ws[t].y == ws[i].y))
         ws[i].killed=1;
}

void findcdot()
{
  int t, i;

  for (t=0; t<dotidx; t+=4)
    for (i=0; i<dotidx; i+=4)
    {
      if ((dot[t+0].y > dot[i+1].y) &&
          (dot[t+1].y < dot[i+1].y) &&
          (dot[t+0].x > dot[i+1].x) &&
          (dot[t+0].x < dot[i+2].x))
         {
            cdot[crsidx].x=dot[t+0].x;
            cdot[crsidx].y=dot[i+1].y;
            crsidx++;
         }

      // 2nd
      if ((dot[t+0].y > dot[i+0].y) &&
          (dot[t+1].y < dot[i+0].y) &&
          (dot[t+0].x > dot[i+0].x) &&
          (dot[t+0].x < dot[i+3].x))
         {
            cdot[crsidx].x=dot[t+0].x;
            cdot[crsidx].y=dot[i+0].y;
            crsidx++;
         }

      // 3rd
      if ((dot[t+3].y > dot[i+1].y) &&
          (dot[t+2].y < dot[i+1].y) &&
          (dot[t+3].x > dot[i+1].x) &&
          (dot[t+3].x < dot[i+2].x))
         {
            cdot[crsidx].x=dot[t+3].x;
            cdot[crsidx].y=dot[i+1].y;
            crsidx++;
         }
```

```
      // 4th
      if ((dot[t+3].y > dot[i+0].y) &&
          (dot[t+2].y < dot[i+0].y) &&
          (dot[t+3].x > dot[i+0].x) &&
          (dot[t+3].x < dot[i+3].x))
         {
            cdot[crsidx].x=dot[t+3].x;
            cdot[crsidx].y=dot[i+0].y;
            crsidx++;
         }
    }
}

void makews()
{
  int t;

  for (t=0; t<dotidx; t++)
    if (!dot[t].killed)
    {
      ws[wsidx].killed=0;
      ws[wsidx].hlink=-1;
      ws[wsidx].vlink=-1;
      ws[wsidx].x=dot[t].x;
      ws[wsidx].y=dot[t].y;
      wsidx++;
    }

  for (t=0; t<crsidx; t++)
    if (!cdot[t].killed)
    {
      ws[wsidx].killed=0;
      ws[wsidx].hlink=-1;
      ws[wsidx].vlink=-1;
      ws[wsidx].x=cdot[t].x;
      ws[wsidx].y=cdot[t].y;
      wsidx++;
    }
}

void prepare()
{
  int t, cnt=0;

  for (t=0; t<wsidx; t++)
    if (!ws[t].killed)
    {
      bws[cnt].killed=0;
      bws[cnt].hlink=-1;
      bws[cnt].vlink=-1;
      bws[cnt].x=ws[t].x;
      bws[cnt].y=ws[t].y;
      cnt++;
```

```
    }

  wsidx=cnt;
  memcpy(&ws, &bws, sizeof(DOTINFO)*MAXDOT*2);
}

void quicksort(int left, int right, int cas)
{
  int l=left, r=right+1, tmp;

  if (l < r)
  {
        switch(cas)
        {
          case 0:tmp=bws[left].x; break;
          case 1:tmp=bws[left].y; break;
        }

        while (1)
        {
          switch(cas)
          {
            case 0:
                do l++; while (bws[l].x < tmp);
                do r--; while (bws[r].x > tmp);
                break;
            case 1:
                do l++; while (bws[l].y < tmp);
                do r--; while (bws[r].y > tmp);
                break;
          }

          if (l < r) swapdata(&bws[l], &bws[r]);
          else break;
        }

        swapdata(&bws[left], &bws[r]);

        quicksort(left, r-1, cas);
        quicksort(r+1, right, cas);
  }
}

void swapdata(DOTINFO *a, DOTINFO *b)
{
  DOTINFO c;

  memcpy(&c, a, sizeof(DOTINFO));
  memcpy(a, b, sizeof(DOTINFO));
  memcpy(b, &c, sizeof(DOTINFO));
}

void link_horizontal()
```

```
{
  int t, np, posa, posb;

  memcpy(&bws, &ws, sizeof(DOTINFO)*MAXDOT*2);
  quicksort(0, wsidx-1, 0);

  for (t=0; t<wsidx; t++)
    if (bws[t].hlink == -1)
    {
      np=findRnearest(t);
      posa=bws2ws(t);
      posb=bws2ws(np);

      bws[t].hlink=np;
      bws[np].hlink=t;

      ws[posa].hlink=posb;
      ws[posb].hlink=posa;
    }
}

int bws2ws(int bwsn)
{
  int t;

  for (t=0; t<wsidx; t++)
    if ((ws[t].x == bws[bwsn].x) &&
        (ws[t].y == bws[bwsn].y)) return t;

  return -1;
}

int findRnearest(int num)
{
  int t;

  for (t=0; t<wsidx; t++)
    if ((bws[t].x > bws[num].x) && (bws[t].y == bws[num].y))
      return t;

  return -1;
}

void link_vertical()
{
  int t, np, posa, posb;

  memcpy(&bws, &ws, sizeof(DOTINFO)*MAXDOT*2);
  quicksort(0, wsidx-1, 1);

  for (t=0; t<wsidx; t++)
    if (bws[t].vlink == -1)
    {
```

```
      np=findBnearest(t);
      posa=bws2ws(t);
      posb=bws2ws(np);

      bws[t].vlink=np;
      bws[np].vlink=t;

      ws[posa].vlink=posb;
      ws[posb].vlink=posa;
    }
}

int findBnearest(int num)
{
  int t;

  for (t=0; t<wsidx; t++)
    if ((bws[t].y > bws[num].y) && (bws[t].x == bws[num].x))
      return t;

  return -1;
}

int notvisit()
{
  int t;

  for (t=0; t<wsidx; t++)
    if (visit[t] == 0) return t;

  return -1;
}

void showlist()
{
  int t, sp, cp;

  printf("SKYLINE:\n");
  while ((sp=notvisit()) != -1)
  {
    cp=sp;
    while (1)
    {
      visit[cp]=1;
      printf("X: %d, Y: %d\n", ws[cp].x, ws[cp].y);
      cp=ws[cp].vlink;
      if (cp == sp) break;

      visit[cp]=1;
      printf("X: %d, Y: %d\n", ws[cp].x, ws[cp].y);
      cp=ws[cp].hlink;
      if (cp == sp) break;
    }
```

```
  }
}

void readdata(char *name)
{
  FILE *f;
  int x1,y1, x2,y2, x3,y3, x4,y4;
  int status;

  if ((f=fopen(name, "rt")) == NULL)
  {
    printf("File reading error.\n");
    exit(1);
  }

  while (1)
  {
    status=fscanf(f, "%d %d %d %d %d %d %d %d\n", &x1, &y1, &x2, &y2, &x3, &y3, &x4, &y4);
    if (status == EOF) break;

    dot[dotidx].x=x1;     dot[dotidx].y=y1;
    dot[dotidx+1].x=x2;   dot[dotidx+1].y=y2;
    dot[dotidx+2].x=x3;   dot[dotidx+2].y=y3;
    dot[dotidx+3].x=x4;   dot[dotidx+3].y=y4;
    dotidx+=4;
  }

  fclose(f);
}

void drawhline(int fx, int fy, int tx, int ty)
{
  int t;

  if (tx < fx)
  {
    t=tx;
    tx=fx;
    fx=t;
  }

  for (t=fx; t<=tx; t++) putdot(t, fy, 15);
}

void drawvline(int fx, int fy, int tx, int ty)
{
  int t;

  if (ty < fy)
  {
    t=ty;
    ty=fy;
    fy=t;
```

```
  }

  for (t=fy; t<=ty; t++) putdot(fx, t, 15);
}

void drawoutgrp()
{
  int t, sp, cp, fx, fy;

  asm mov ax, 13h
  asm int 10h

  for (t=0; t<MAXDOT*2; t++) visit[t]=0;

  while ((sp=notvisit()) != -1)
  {
    cp=sp;
    while (1)
    {
      visit[cp]=1;
      fx=ws[cp].x;
      fy=ws[cp].y;
      cp=ws[cp].vlink;
      delay(100);
      drawvline(fx, fy, ws[cp].x, ws[cp].y);

      if (cp == sp) break;

      visit[cp]=1;
      fx=ws[cp].x;
      fy=ws[cp].y;
      cp=ws[cp].hlink;
      delay(100);
      drawhline(fx, fy, ws[cp].x, ws[cp].y);

      if (cp == sp) break;
    }
  }

  getch();

  asm mov ax, 3h
  asm int 10h
}

void drawdatagrp()
{
  int t;

  asm mov ax, 13h
  asm int 10h

  for (t=0; t<dotidx; t+=4)
```

```
  {
     drawvline(dot[t].x, dot[t].y, dot[t+1].x, dot[t+1].y);
     drawhline(dot[t+1].x, dot[t+1].y, dot[t+2].x, dot[t+2].y);
     drawvline(dot[t+2].x, dot[t+2].y, dot[t+3].x, dot[t+3].y);
     drawhline(dot[t+3].x, dot[t+3].y, dot[t].x, dot[t].y);
  }

  getch();

  asm mov ax, 3h
  asm int 10h
}
```