Buletinul Ştiinţific al Universităţii din Baia Mare
Seria B, Matematică-Informatică, vol.X(1994),99-107

AN ELEMENTARY GEOMETRIC CONSTRUCTION PERFORMED BY A COMPUTER

Vasile BERINDE and Ovidiu COSMA

The aim of this note is to solve and extend an elementary geometric construction problem using a computer. The obtained alghorithm and the corresponding program can be used in computer aided lessons.

P1. Construct, with the aid of a pair of compasses and a ruler, a triangle having the same area as a given convex quadrilateral.

(Problem 4, page 14, [2])

Solution. Let ABCD be the given quadrilateral. We draw a diagonal, say AC and then we draw a parallel d to this diagonal, passing by D. For each M∈d we have

$$Area\ of\ DAC = Area\ of\ MAC\ .$$

Based on this property, the construction follows immediately: let E be the point of intersection of BC and d. Then BEA is the required triangle.

Remark. The convexity hypothesis of the quadrilateral is essential in the solution of problem P1. It is also easy to see that the construction can by iteratively applied for a convex pentagon: in the first step we construct a convex quadrilateral having the same area as the given pentagon and then, using P1 we obtain the desired triangle. In this case it was necessary to draw two diagonals in the given pentagon.

In the same way we can solve a more general problem

P2. Construct with the aid of a pair of compasses and a ruler a triangle having the same area as a given convex polygon.

Solution. Let $A_1A_2...A_n$ be the given convex polygon, $n\geq 4$. We draw the diagonal A_1A_3 and a parallel d_1 to A_1A_3, passing by A_2. For each $M\in(d_1)$ we have

$$Area\ of\ A_1A_2A_3 = Area\ of\ MA_1A_3\ .$$

Let E_1 be the point of intersection of d_1 and A_3A_4; then the area of the polygon $A_1E_1A_4...A_n$ equals the area of the given convex polygon.

Now, if n=4, the problem is completely solved. Otherwise, we draw the diagonal A_1A_4 and a parallel d_2 to A_1A_4, passing by E_1, and so on. The claimed triangle is obtained when we can't draw a diagonal, that is when we just finished to do the corresponding construction to the diagonal A_2A_{n-1}. In this case, if E_{n-3} is the point of intersection of d_{n-2} and A_nA_{n-1}, then the triangle $A_1E_{n-3}A_n$ is the required triangle, i.e.

$$Area\ of\ A_1A_2...A_n = Area\ of\ A_1E_{n-3}A_n\ .$$

Remark. This alghoritm may be easely programmed on a computer in order to be used for computer aided lessons. Here is the text of the program written in C language on an IBM PC compatible computer. Finally, the construction for n=5 is given.

```
// EQUIVALENT TRANSFORMATIONS OF POLYGONS
// -------------------------------------
// initial polygons coordinates are taken from file "FIGURI",
// ,in the following format:
// $,n,x1,y1,x2,y2,x3,y3 ...
// $ indicates beginning of figure data;
// n = number of corners of the polygon;
// xi, yi = corner coordinates. The file can hold maximum 20 figures.

#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
```

```
struct pt{
        float x;
        float y;
};
void despol(int, struct pt[]);
void blink(int, int, struct pt[], int);
void show_line(int, int, int, int, int, int);
void show_circle(int, int, int, int, int);
void endsound(int);

struct pt inter[2], inter2[2];
int maxx, maxy;
int nr_int;
int main(void)
{
    FILE *in;
    int figuri[20], cda, j;
    int i, n, t;
    float xk, yk, x, y, x1, x2, y1, y2, x3, y3, x4, y4, a, b, c, d, m;
    struct pt pol[10], tri1[3];
    int gdriver = DETECT, gmode, errorcode;
    initgraph(&gdriver, &gmode, "");
    errorcode = graphresult();
    if (errorcode != grOk)
    {
        printf("Graphic mode error: %s\n", \
                 grapherrormsg(errorcode));
        printf("Press any key");
        getch();
        exit(1);
    }
    maxx = getmaxx();
    maxy = getmaxy();

    // main loop
    // ------------------------------
    while(1){
        setbkcolor(BLACK);
        clrscr();
        cleardevice();
                    // open file "figuri"
        if ((in = fopen("figuri", "rt")) == NULL)
        {
            fprintf(stderr, "Read error \n");
            return 1;
        }

                    //search all the figures in the file
        i=0;
        while (!feof(in)){
            if (fgetc(in)=='$')
                fscanf(in, ",%d", &figuri[i++]);
        };
        printf("\n\nThe file contains the following figures:");
```

```
for(j=0; j<i; j++)
    printf("\n(%d)  n = %d", j+1, figuri[j]);
        //  let the operator choose a figure
printf("\n\nIndicate the figure by pressing the associated number \
        \nN=No. of figure / 0=Exit\n");
scanf("%d", &cda);
if(cda==0)
    break;
            //get the selected figure data
j=0;
fseek(in,0,SEEK_SET);
while (!feof(in) && j!=cda){
    if (fgetc(in)=='$')
        j++;
};
fscanf(in, ",%d,", &n);
for(i=0,j=0; i<n; i++){
    fscanf(in, "%d,", &t); pol[j].x=t;
    fscanf(in, "%d,", &t); pol[j++].y=t;
}
fclose(in);

// display the selected figure equivalent transformations
//-------------------------------
do{
            // show the original polygon
    setfillstyle(SOLID_FILL,BLUE);
    setbkcolor(LIGHTGRAY);
    cleardevice();
    despol(n, pol);
    getch();

            // loop for subsequent transformations
            // unify points 0 and n-2
    show_line(pol[0].x, pol[0].y, pol[n-2].x, pol[n-2].y,\
    DARKGRAY, WHITE);

            // draw a parallel to the previous line
    m=(pol[0].y-pol[n-2].y)/(pol[0].x-pol[n-2].x);
    nr_int=0;
    xk=pol[n-1].x; yk=pol[n-1].y;
    y=yk+m*(-xk);
    if(y>=0 && y<=maxy){
        inter[nr_int].x=0; inter[nr_int++].y=y;
    }
    y=yk+m*(maxx-xk);
    if(y>=0 && y<=maxy){
        inter[nr_int].x=maxx; inter[nr_int++].y=y;
    }
    x=(-yk)/m+xk;
    if(x>=0 && x<=maxx && nr_int <2){
        inter[nr_int].x=x; inter[nr_int++].y=0;
    }
    x=(maxy-yk)/m+xk;
```

```
if(x>=0 && x<=maxx && nr_int <2){
    inter[nr_int].x=x; inter[nr_int++].y=maxy;
}

setlinestyle(SOLID_LINE,1,1);
show_line(inter[0].x, inter[0].y, inter[1].x, inter[1].y, \
DARKGRAY, YELLOW);
        //draw the line that passes through the points n-3 and n-2
x1=pol[n-3].x; y1=pol[n-3].y;
x2=pol[n-2].x; y2=pol[n-2].y;
nr_int=0;
y=-x1*(y2-y1)/(x2-x1)+y1;
if(y>=0 && y<=maxy){
    inter2[nr_int].x=0; inter2[nr_int++].y=y;
}
y=(maxx-x1)*(y2-y1)/(x2-x1)+y1;
if(y>=0 && y<=maxy){
    inter2[nr_int].x=maxx; inter2[nr_int++].y=y;
}
x=-y1*(x2-x1)/(y2-y1)+x1;
if(x>=0 && x<=maxx && nr_int <2){
    inter2[nr_int].x=x; inter2[nr_int++].y=0;
}
x=(maxy-y1)*(x2-x1)/(y2-y1)+x1;
if(x>=0 && x<=maxx && nr_int <2){
    inter2[nr_int].x=x; inter2[nr_int++].y=maxy;
}

show_line(inter2[0].x, inter2[0].y, inter2[1].x, inter2[1].y, \
DARKGRAY, YELLOW);
        // find the intersection coordinates and store them in x,y
x1=inter[0].x; y1=inter[0].y; x2=inter[1].x; y2=inter[1].y;
x3=inter2[0].x; y3=inter2[0].y; x4=inter2[1].x; y4=inter2[1].y;
a=y2-y1; b=x2-x1; c=y4-y3; d=x4-x3;
x=(b*(d*y3-c*x3)+d*(a*x1-b*y1))/(a*d-c*b);
y=a*(x-x1)/b+y1;
        // show intersection
setlinestyle(SOLID_LINE,1,3);
show_circle(x, y, 5, DARKGRAY, LIGHTMAGENTA);
        // show the 2 equivalent triangles
setfillstyle(EMPTY_FILL,1);
        // first
tri1[0].x=pol[0].x;   tri1[0].y=pol[0].y;
tri1[1].x=pol[n-2].x; tri1[1].y=pol[n-2].y;
tri1[2].x=pol[n-1].x; tri1[2].y=pol[n-1].y;
blink(DARKGRAY, LIGHTGREEN, tri1, 3);
                // second
tri1[0].x=pol[0].x;   tri1[0].y=pol[0].y;
tri1[1].x=pol[n-2].x; tri1[1].y=pol[n-2].y;
tri1[2].x=x;          tri1[2].y=y;
blink(DARKGRAY, LIGHTRED, tri1, 3);
        // continues the transformation with n-1 points
n--;
        // move the point n-1 in the intersection of the 2 lines
```

```
            pol[n-1].x=x; pol[n-1].y=y;
                    // show the equivalent polygon
            blink(DARKGRAY, WHITE, pol, n);
            cleardevice();
            setcolor(WHITE);
            sound(1500);
            delay(50);
            nosound();
        }
            while(n>3);
            // show the final polygon
        setfillstyle(SOLID_FILL,BLUE);
        despol(n, pol);
        getch();
        cleardevice();
        endsound(60);
    }
                // clean up */
    closegraph();
    return 0;
}
void show_circle(int x, int y, int r, int color1, int color2){
//------------------------------
        while(1){
            setcolor(color1);
            circle(x,y,r);
            delay(200);
            setcolor(color2);
            circle(x,y,r);
            delay(200);
            if(kbhit()){
                getch();
                break;
            }
        }
}
void show_line(int x1, int y1, int x2, int y2, int color1, int color2){
//------------------------------
    while(1){
        setcolor(color1);
        line(x1, y1, x2, y2);
        delay(200);
        setcolor(color2);
        line(x1, y1, x2, y2);
        delay(200);
        if(kbhit()){
            getch();
            break;
        }
    }
}
```

```
void blink(int color1, int color2, struct pt fig[], int n){
//------------------------------
    while(1){
        setcolor(color1);
        despol(n, fig);
        delay(200);
        setcolor(color2);
        despol(n, fig);
        delay(200);
        if(kbhit()){
            getch();
            break;
        }
    }
}

void despol(int n, struct pt p[]){
//------------------------------
                // prepare the corners
    int poly[20];
    int i,j;
    for(i=0,j=0; i<n; i++){
        poly[j++]=p[i].x;
        poly[j++]=p[i].y;
    }
                // close the polygon
    poly[j++]=p[0].x;
    poly[j]=p[0].y;
                // draw the figure
    setlinestyle(SOLID_LINE,1,3);
    fillpoly(n+1, poly);
}
void endsound(int i){
//------------------------------
    sound(800);
    delay(i);
    nosound();
    delay(i);
    sound(1500);
    delay(i);
    nosound();
    delay(i);
    sound(2000);
    delay(i);
    nosound();
}
```

Pasul 1.1.

Pasul 1.2.

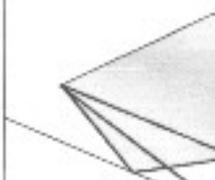
Pasul 1.3.

Pasul 1.4.

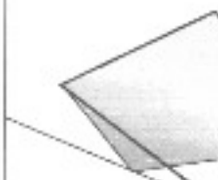
Pasul 1.5.

Pasul 1.6.

Pasul 1.7.



Pasul 1.8.



Pasul 2.1.

Pasul 2.2.

Pasul 2.3.

Pasul 2.4.

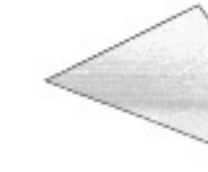
Pasul 2.5.

Pasul 2.6.

Pasul 2.7.

Pasul 2.8.

Pasul 3.1.



REFERENCES

1.BERINDE,V., How to stimulate the engineering invention by means of mathematical approaches, Proceedings of the European Conference "Teaching Mathematics for Industry", September 18-20, 1994, Czech Technical University, Prague

2.COŢA,A., Geometrie şi trigonometrie, Manual pentru clasa a X-a, Editura Didactică şi Pedagogică, Bucureşti, 1983.

Received: April 19, 1994

UNIVERSITY OF BAIA MARE
DEPARTMENT OF MATHEMATICS
4800 BAIA MARE
ROMANIA