Térbeli alakzatok ábrázolása számítógéppel

SZILASSI LAJOS

Manapság – akarva-akaratlanul – szemlélői vagyunk a TV képernyőjén pergő-forgó kockák és egyéb alakzatok látványának. Ezt a néhány éve még elképzelhetetlen, ötletdús szín- és formagazdag látványt, a mozgás illúziójának ezt a szellemes, változatos bemutatását a mai számítógépek gyorsaságának, információtároló kapacitásának és főleg az erre támaszkodó képalkotó, képfeldolgozó programoknak köszönhetjük. Ezeket a programokat nagy szoftverházak sok-sok munkatársa összehangolt teammunkával állítja elő, így egyetlen – bármilyen jól felkészült – programozó sem vállalkozhat arra, hogy közel hasonló színvonalú programot készítsen. Legfeljebb arra vállalkozhatunk, hogy megpróbáljuk megérteni e nagy programrendszerek matematikai hátterét azzal, hogy készítünk egy szerény, térgeometriai alakzatokat ábrázoló programot, amelynek minden részletét, főleg azok matematikai vonatkozásait ismerjük, értjük. Ezzel egyúttal lehetőségünk nyílhat arra, hogy ne csak a mások által kitalált alakzatokat szemlélgessük, hanem magunk is előállíthatunk olyan alakzatokat, amelyeket matematikai módszerekkel jól le tudunk írni.

Matematikai háttér

Lényegében egyetlen kulcskérdést kell alaposan végiggondolnunk: miként lehet egyetlen – térbeli koordinátáival adott – pontnak a síkra (a képernyő síkjára) eső vetületét, azaz e vetületnek a képernyő koordinátarendszerében vett koordinátáit előállítani. Ehhez először pontosan meg kell határoznunk a térbeli koordinátarendszernek a képernyő síkjához viszonyított helyzetét, majd megadnunk e térbeli koordinátarendszer egységvektorainak a kiválasztott vetítési módszerrel kapott képét, (pontosabban ezen egységvektorok vetületeinek a síkbeli koordinátarendszerben vett koordinátáit). Ezzel lényegében megadjuk azokat a transzformációs képleteket, amelyekkel a tér egy EMBED Equation pontjához hozzárendeljük ennek a képernyőre eső EMBED Equation vetületét.

Ezt a vetületet természetesen többféle vetítési módszerrel is előállíthatjuk. Akik a rajz (mint tantárgy, vagy mint képzőművészet) irányából közelítik meg a problémát, előnyben részesítik az ún. *perspektív* ábrázolást, amely a tér egy adott pontjára – a centrumra – illeszkedő vetítősugarakkal vetíti a tárgyat a képsíkra, jelen esetben a képernyő síkjára. A perspektív ábrázolással többnyire az a probléma, hogy ha nem az ábrázolás centrumából nézzük a képet, akkor torznak látjuk. Különösen, ha a kép készítésekor közelebb volt a tárgyhoz a vetítés centruma, mint a mi nézőpontunk. Ezért még a szemléletesség szempontjából is, de főként a szerkesztés (vagy jelen esetben a számolás) szempontjából előnyösebb az ún. *axonometrikus* ábrázolás, amely a perspektív ábrázolás olyan speciális esetének is tekinthető, amelyben a centrum végtelen távoli, azaz a vetítősugarak párhuzamosak. A különböző axonometrikus ábrázolási módok közül az *ortogonális* (merőleges) *axonometria* adja a legszemléletesebb képet, amelynek a lényege, hogy az

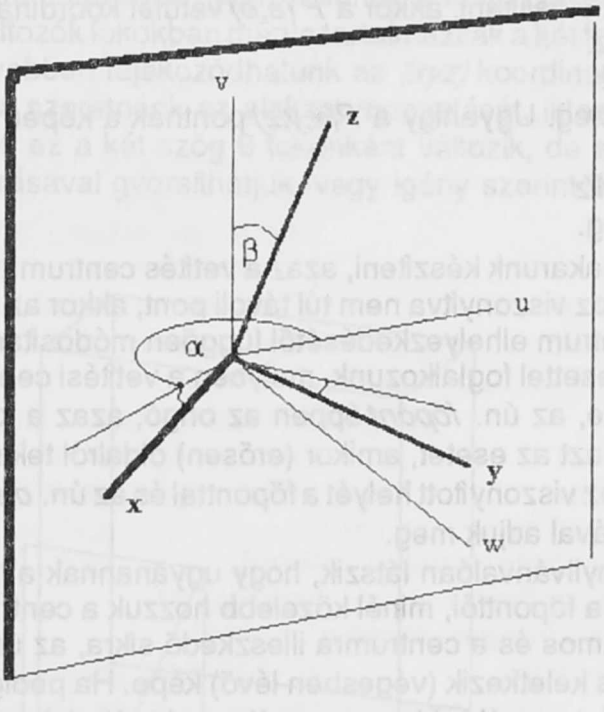
ábrázolandó alakzatot a képsíkhoz viszonyítva a lehető legáltalánosabb helyzetben elhelyezve a képsíkra merőleges irányból végezzük a vetítést.

Most egy ilyen ortogonális axonometrikus, illetve perspektív kép készítésére látunk példát. A mellékelt – PASCAL nyelven írt – program alkalmazása jó lehetőséget kínál arra, hogy összehasonlíthassuk a kétféle ábrázolási módot. Remélhetőleg az itt leírt eljárások lényegét azok is megértik, akiknek nincs kellő jártasságuk a PASCAL nyelvben. A gyakorlottabbak, vagy az éppen programozni tanulók részére viszont izgalmas lehetőség ennek a programnak az elemzése, saját elképzeléseik szerinti továbbfejlesztése.

A (mozgó) térbeli koordinátarendszer tengelyei legyenek rendre *x, y* és *z*, a képernyőhöz képest rögzített (egyelőre ugyancsak térbelinek tekintett) koordinátarendszer tengelyei pedig rendre *u, v* és *w*. A két koordinátarendszer origója legyen közös, pl. a képernyő $O(x_o,y_o)$ középpontja. Állítsuk be egyelőre az *(xyz)* rendszert úgy, hogy a *z* tengelye függőlegesen, az *x* vízszintesen, jobbfelé álljon. Ha rendszerünk jobbsodrású, akkor az *y* tengely a képernyő síkjának a mi nézőpontunkkal ellentétes féltere felé mutat, arra merőleges irányba. A képernyőhöz képest rögzített *(uvw)* rendszer *u* tengelye essen egybe az *x*-szel, *v* a *z*-vel. Ha ez is jobbsodrású *u, v, w* sorrendben, akkor most *w* az *y*-nal ellentétes irányba, tehát felénk mutat. (Később látni fogjuk, hogy a perspektív kép készítéséhez szükségünk lesz a térbeli *P(x,y,z)* pontnak a képernyőtől mért távolságára, ezért kellett felvennünk a *w* tengelyt is.)

A feladatunk az lesz, hogy az *(xyz)* koordináterendszert mozgatva meghatározzuk e rendszer egységvektorainak az *(uvw)* rendszerbeli koordinátáit, amelyeket rendre jelöljön *Ux, Vx, Wx, Uy, Vy, Wy* ill. *Uz, Vz, Wz*.

Először forgassuk el az így beállított koordinátarendszert a függőleges *v (=z)* tengely körül α szöggel, majd az *(xy)* sík és a képsík metszésvonala (vagyis a – rögzített – *u* tengely) körül ß-val. Ezzel a két forgatással elő tudjuk állítani az *(xyz)* rendszer minden olyan helyzetét, amelyben a *z* tengely illeszkedik egy az *u*-ra merőleges (tehát függőleges) síkra, így a képsíkra eső merőleges vetülete is függőleges egyenes, vagy egyetlen pont lesz. Egy újabb, (pl. a képernyő síkjára merőleges, az origóra illeszkedő tengely körüli) forgatással az is elérhető, hogy a *z* tengely képe ne maradjon függőleges helyzetű. Ezzel a térbeli koordináterendszer valóban tetszőlegesen állhat a képsíkhoz képest. (E legutóbbi művelet elvégzését az olvasóra bízzuk.)



Az α szögű, *z* tengelyű forgatást végrehajtva kapjuk, hogy

$Ux = \cos(\alpha)$, $Vx = 0$, $Wx = -\sin(\alpha)$

Mivel az *y* tengely az *x* tengelynek (a *z* tengely pozitív felét tartalmazó féltérből nézve) $+90^{\circ}$-os elforgatottja, ezért

$Uy = \cos(\alpha+90^\circ)$, $Vy = 0$, $Wy = -\sin(a+90o)$.

Azaz

$Uy = \sin(\alpha)$, $Vy = 0$, $Wy = -\cos(\alpha)$.

Mivel z volt a forgatás tengelye, a z irányú egységvektor egyelőre nem mozdult:

$Uz = 0$, $Vz = 1$, $Wz = 0$.

(Wx és Wy azért kapott negatív előjelet, mert a w tengely pozitív fele az u pozitív felével 270°-os – vagy, ha úgy tetszik, $-90o$-os – szöget zár be.)

Ahhoz, hogy ráláthassunk az (xy) síkra, döntsük most magunk felé az (xyz) rendszert, azaz forgassuk el az u tengely körül ß szöggel. (Akkor tekintsük ß-t pozitívnak, ha $0<ß<180^\circ$-os forgatás után a z tengely pozitív fele a képernyő síkjának a felénk eső térfelébe esik E forgás során minden pont egy olyan körívet ír le, amelynek a síkja merőleges az u tengelyre, így Ux, Uy, Uz nem változik.

E forgás során minden pont egy olyan körívet ír le, amelynek a síkja merőleges az u tengelyre, így Ux, Uy, Uz nem változik.

Ha egy ilyen forgást végző pont a forgatást megelőzően az u tengelytől c távolságra volt, akkor a forgatás után $c*cos(ß)$ lesz a v irányú, $c*sin(ß)$ a w irányú koordinátája. Így, mivel az (xy) sík $ß+90^\circ$-os szöget zár be a képernyő síkjával,

$Vx = Wx\cos(ß+90^\circ) = (-\sin(\alpha)\ (-\sin(ß)) = \sin(\alpha)*\sin(ß)$,

$Vy = Wy\cos(ß+90^\circ) = (-\cos(\alpha)\ (-\sin(ß)) = \cos(\alpha)*\sin(ß)$,

$Vz = \cos(ß)$.

Ugyanígy:

$Wx = Wx\sin(ß+90^\circ) = \sin(\alpha)\cos(ß)$,

$Vy = Wy\sin(ß+90^\circ) = \cos(\alpha)\cos(ß)$,

$Wz = \sin(ß)$.

Meg kell még változtatnunk a V_x, V_y, V_z koordináták előjelét, mivel a képernyő koordinátarendszerében az ordinátatengely lefelé mutat.

Összefoglalva tehát a az alábbi transzformációs képletek írják le az (yxz) koordinátarendszer egységvektorainak az (uvw) rendszerbeli koordinátáit:

$Ux = \cos\alpha$, $Vx = -\sin(\alpha)\sin(ß)$, $Wx = -\sin(\alpha)\cos(ß)$,

$Uy = -\sin\alpha$, $Vy = -\cos(\alpha)\sin(ß)$, $Wy = -\cos(\alpha)\cos(ß)$,

$Uz = 0$, $Vz = -\cos(ß)$, $Wz = \sin(ß)$.

Ha a $P(x,y,z)$ pontnak a képernyőre eső merőleges vetületét azaz ortogonális axonometrikus képét akarjuk előállítani, akkor a $P'(a,b)$ vetület koordinátáit egyszerűen az

$a = x*Ux+y*Uy+z*Uz$,

$b = x*Vx+y*Vy+z*Vz$

képletekkel kapjuk meg. Ugyanígy a $P(x,y,z)$ pontnak a képernyő síkjától mért távolságát a

$c = x*Wx+y*Wy+z*Wz$

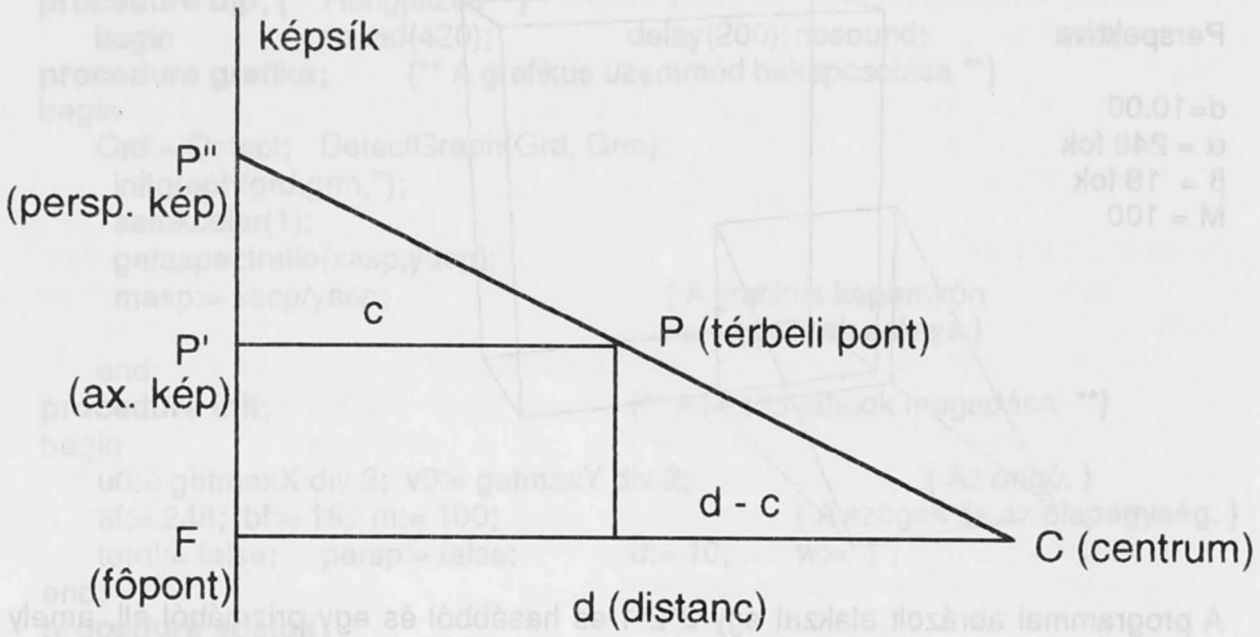
képlet határozza meg.

Ha perspektív képet akarunk készíteni, azaz a vetítés centruma egy a képernyőhöz ill. az ábrázolandó tárgyhoz viszonyítva nem túl távoli pont, akkor az előbbi $P'(a,b)$ pont helyét a képernyőn a centrum elhelyezkedésétől függően módosítanunk kell.

Most csak azzal az esettel foglalkozunk, melyben a vetítési centrumnak a képernyőre eső merőleges vetülete, az ún. *főpont* éppen az origó, azaz a képernyő középpontja, vagyis nem vizsgáljuk azt az esetet, amikor (erősen) oldalról tekintünk a képernyőre. A centrumnak a képsíkhoz viszonyított helyét a főponttal és az ún. *distanccal*, azaz a képsík és a centrum távolságával adjuk meg.

A mellékelt ábráról nyilvánvalóan látszik, hogy ugyanannak a térbeli pontnak a képe annál messzebb kerül a főponttól, minél közelebb hozzuk a centrumot a képsíkhoz.

A képsíkkal párhuzamos és a centrumra illeszkedő síkra, az ún. *eltűnési síkra* illeszkedő pontoknak nem is keletkezik (végesben lévő) képe. Ha pedig egy pont és a képsík távolsága nagyobb a distancnál, akkor a pontról un. virtuális képet kapunk. Ezt az esetet csak úgy kerülhetjük el, ha a centrumot a képsíktól távolabb helyezzük el, mint a pontok koordinátáinak a maximuma.

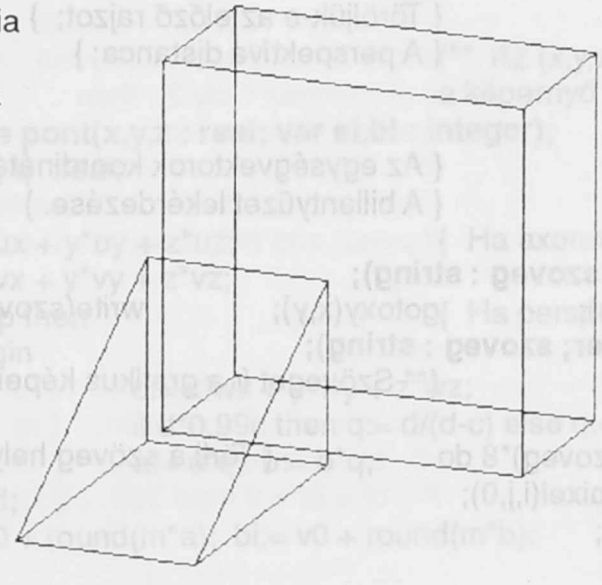


Az ábra szerint $\frac{P''F}{P'F} = \frac{d}{d-c}$ ami azt jelenti, hogy a *P'(a,b)* axonometrikus képből a koordináták $q = \frac{d}{d-c}$ szeres nyújtásával kapjuk a P'' perspektív kép koordinátáit. (Ebből a képletből is látszik, hogy miért nem engedhető meg a d=c eset.)

A mellékelt programot itt talán nem szükséges részletesen elemezni. A fentiekből kitűnik, hogy a program kulcsfontosságú eljárása a *vektorok* nevű, amely az *Ux, Uy, ... Wz* globális változók aktuális értékeit adja. Ebből azután a *pont* nevű eljárás készíti el egy adott pontnak a képernyő koordinátarendszerében vett koordinátáit. Ezt pedig a *szakasz* nevű eljárás hívja, amely két *(xyz)* rendszerbeli pont összekötő szakaszát rajzolja a képernyőre. (Tulajdonképpen ez az egyetlen output eljárás a programban.) A program többi része lényegében a fenti eljárások meghívásának a megszervezéséből áll. Az *af* és *bf* ugyancsak globális változók fokokban mérve tartalmazzák a két forgatás szögét, így ezek értékeit kiíratva könnyebben tájékozódhatunk az *(xyz)* koordinátarendszer helyzetéről. Ha azonban gyorsítani szeretnénk az alakzat mozgatását, iktassuk ki az *adatok* nevű eljárást. Jelen esetben ez a két szög 6 fokonként változik, de a *fut* nevű eljárásban a megfelelő számok átírásával gyorsíthatjuk, vagy igény szerint finomíthatjuk az alakzat mozgását.

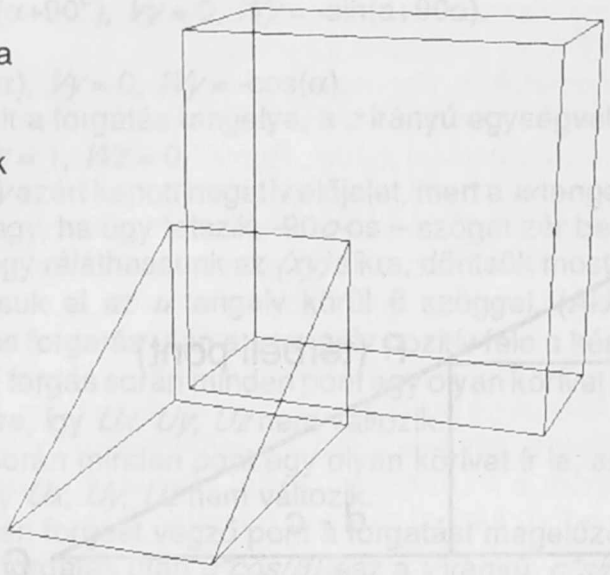
Axonometria

α = 248 fok
β= 18 fok
M = 100



Perspektíva

d=10.00
α = 248 fok
β = 18 fok
M = 100



A programmal ábrázolt alakzat egy 2*2*1-es hasábból és egy prizmából áll, amely együttes sem tengelyesen sem centrálisan nem szimmetrikus, így segítségével jól szemlélhető ábrákhoz jutunk annak ellenére, hogy most nem foglalkoztunk a láthatóság feltüntetésével, amely az eddigieknél sokkal nehezebb probléma mind matematikai, mind programozástechnikai szempontból.

A következő két rajz a program kezdő paraméterértékeivel készült axonometrikus ill. perspektív kép.

A program

```
program axonometria;
uses crt, graph;
var                                   {** Általános grafikai változók: **}
    grd, grm        :integer;         { A képernyő vizsgálata, bekapcsolása }
    xasp, yasp      :word;
    masp            :real;            { A tengelyekre eső egységek aránya }
                                      {** Ennek a programnak
                                      a globális változói: **}
    u0, v0          :integer;         { A koordinátarendszer középpontja; }
    m               :integer;         { Az alapegység;}
    af, bf          :integer;         { af: az X tengely és a képsík szöge
                                      (ha a Z tengely a képsíkban van),
                                       bf: a Z tengely
                                      és a képsík szöge (fokokban) }
    persp           :boolean;         { Perspektíva vagy axonometria; }
    torol           :boolean;         { Töröljük-e az előző rajzot; }
    d               :real;            { A perspektíva distanca; }
    ux, vx, wx,
    uy, vy, wy,
    uz, vz, wz      :real;            { Az egységvektorok koordinátái; }
    w               :char;            { A billentyűzet lekérdezése. }

procedure iras(s,x,y : byte; szoveg : string);
    begin      textcolor(s);      gotoxy(x,y);      write(szoveg);    end;
procedure griras(x,y : integer; szoveg : string);
    var i, j          :integer;       {** Szöveget ír a grafikus képernyőre **}
    begin
    for i:=x-5 to x+5+length(szoveg)*8 do          { Törli a szöveg helyét }
     for j:=y-2 to y+10 do putpixel(i,j,0);
       OutTextXY(x,y,szoveg);
end;
```

```
procedure bip; {** Hangjelzés **}
    begin      sound(420);          delay(200); nosound;          end;
procedure grafika;      {** A grafikus üzemmód bekapcsolása **}
begin
    Grd:= Detect;  DetectGraph(Grd, Grm);
     initgraph(grd,grm,'');
     setbkcolor(1);
     getaspectratio(xasp,yasp);
     masp:= xasp/yasp;                    { A grafikus képernyőn
                                            az egységek aránya }
    end;
procedure init;                          {** A kezdő adatok megadása **}
begin
    u0:= getmaxX div 2; v0:= getmaxY div 2;                  { Az origó. }
    af:= 248; bf:= 18; m:= 100;                    { A szögek és az alapegység. }
    torol:= false;   persp:= false;        d:= 10;    w:= 'T';
end;
procedure adatok;
var s : string;
    begin
    if persp then   begin
    str(d:5:2,s); s:= 'Perspektíva: d = ' + s
                    end else  s:= 'Axonometria       ';
    griras(20,30, s);
    str(af:4,s);      s:= chr(224) + ' = ' + s + ' fok';    griras(50,50,s);
    str(bf:4,s);      s:= chr(225) + ' = ' + s + ' fok';    griras(50,70,s);
    str(m :4,s);      s:= 'm = ' + s;                       griras(50,90,s);
end;
procedure vektorok;   {* A térbeli egységvektorok átszámítása}
var  α, beta : real;                     { a síkbeli koordináta-rendszerbe *}
begin
    α:= pi*af/180;  beta:= pi*bf/180;
    ux:=  cos(α);    vx:= -sin(α)*sin(beta)*masp;
    uy:= -sin(α);    vy:= -cos(α)*sin(beta)*masp;
    uz:=  0;         vz:= -cos(beta)*masp;
    if persp then   begin
                    wx:= -sin(α)*cos(beta);
                    wy:= -cos(α)*cos(beta);
                    wz:=  sin(beta);
    end;
end;
                                         {**  Az (x,y,z) térbeli pont képe
                                         a képernyő (ai,bi) pontja **}
procedure pont(x,y,z : real; var ai,bi : integer);
var a, b, c, q  :real;
begin
    a:= x*ux + y*uy + z*uz;              { Ha axonometrikus a kép  }
    b:= x*vx + y*vy + z*vz;
    if persp then                        { Ha perspektív a kép     }
       begin
                    c:= x*wx + y*wy + z*wz;
                    if d*0.99c then q:= d/(d-c) else q:= 100;
                    a:= a*q;  b:= b*q;
       end;
    ai:= u0 + round(m*a);  bi:= v0 + round(m*b);
end;
```

```
procedure szakasz(x1,y1,z1,x2,y2,z2 : real);
var a1,b1,a2,b2 : integer;
begin
    pont(x1,y1,z1,a1,b1);
    pont(x2,y2,z2,a2,b2);
    line(a1,b1,a2,b2);
end;
procedure rajzol;
var  i,j : byte;
begin
    if torol then cleardevice else griras(20,10,'Nem töröl !');
    vektorok;
    setcolor(14);    { Hasáb rajza:  }
    for i :=0 to 1 do
     for j:=0 to 1  do
       begin
         szakasz( i, 2*j,   0, i, 2*j,   2);
         szakasz( i, 0, 2*j, i, 2, 2*j);
         szakasz( 0, 2*i, 2*j, 1, 2*i, 2*j);
       end;
    setcolor(13);                          { Prizma rajza: }
    for i:= 0 to 1 do
     begin
      szakasz(1.2, i, 0, 1.2, i, 1);
      szakasz(1.2, i, 0, 3, i,0);
      szakasz(1.2, i,  1, 3, i, 0);
     end;
    szakasz(1.2, 0, 0, 1.2, 1, 0);
    szakasz(1.2, 0, 1, 1.2, 1, 1);
    szakasz(3, 0, 0, 3, 1, 0);
    setcolor(15);
    adatok;
end;
procedure fut;                             { A kép változtatása és új rajza }
    begin
     repeat
      case w of
                    'T'                    :torol:= not torol;
                    'P'                    :persp:= not persp;
                    'X'                    :begin af := 270; bf := 0; end;
                    'Y'                    :begin af := 180; bf := 0; end;
                    'Z'                    :begin af := 270; bf :=90; end;
                    '+'                    :if m  getmaxY div 2   then
                                            m:= m + 10 else bip;
                    '-'                    :if m  20   then  m:= m  −10  else bip;
                    '*'                    :if persp then d:= d + 0.2 else bip;
                    '/'                    :if persp and (d4) then  d:= d-0.2  else bip;
                    chr(0)                 :begin
                                            w:=readkey;
                                            case w of
                                            'K'  : af:= af – 6 mod 360;   { fel  }
                                            'M' : af:= af + 6 mod 360;    { le   }
                                            'H' : bf:= bf – 6 mod 360;    { bal  }
                                            'P' : bf:= bf + 6 mod 360;    { jobb }
    else w:='&' end;
end;
```

```
      else w:='&'; end;
      if w'&' then rajzol else bip;
      if d  30 then begin persp:= false; d:=10; end;
      while keypressed do w:=readkey; w:= upcase(readkey);
    until w=chr(27);
end;
procedure fejlec ;
begin
clrscr;
    iras(13,25,2,'                              Axonometrikus (vagy) perspektív kép');
    iras(11,12,5,'                              A billentyűzet használata:          ');
    iras(11,5, 7, chr(17)+'-  -'+chr(16)+' '+chr(24)+' '+chr(25)
                  +'                            :A lerajzolt alakzat mozgatása;' );
    iras(11,5, 9,'+ , –                         :  A rajz növelése ill. csökkentése; ');
    iras(11,5,11,' X, Y, Z  :                   Merőleges vetületek beállítása. ');
    iras(11,5,13,'                              Ha perspektív rajzot készítünk:');
    iras(11,5,15,' * , /    :                   A distanc növelése ill. csökkentése; ');
    iras(11,5,17,'                              Kétállapotú kapcsolók:');
    iras(11,8,19,'P                             :  Axonometrikus, vagy perspektív kép;  ');
    iras(11,8,21,'T                             :  új rajz készítésekor töröljük-e az előzőt; ');
    iras(12,7,24,'ESC                           :  Kilépés a programból.');
    while not keypressed do;
end;
```