UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA ALGEBRY A GEOMETRIE

BAKALÁŘSKÁ PRÁCE

Využití programu POV-Ray při výuce deskriptivní geometrie

Vedoucí bakalářské práce:
RNDr. Lukáš Rachůnek, Ph.D.
Rok odevzdání: 2011

Vypracoval:
Jan Mlčůch
M-DG, III. ročník

Prohlášení

Prohlašuji, že jsem vytvořil tuto bakalářskou práci samostatně za vedení RNDr. Lukáše Rachůnka, Ph.D. a že jsem v seznamu použité literatury uvedl všechny zdroje použité při zpracování práce.

V Olomouci dne 23. května 2011

Poděkování

Rád bych na tomto místě poděkoval vedoucímu bakalářské práce RNDr. Lukáši Rachůnkovi, Ph.D. za obětavou spolupráci i za čas, který mi věnoval při konzultacích. Dále si zaslouží poděkování typografický systém TEX, kterým je práce vysázena.

Obsah

Úvod

Při výuce deskriptivní geometrie bývá někdy problém představit si, jak zobrazovaná scéna ve skutečnosti vypadá. Je to problém zejména u některých složitějších prostorových scén, kde vystupuje několik objektů. Například průsečík přímky s rovinou, průniky těles atd.

Cílem této práce je ukázat možnosti programu POV-Ray při tvorbě doplňujících obrázků pro výuku deskriptivní geometrie. Snažil jsem se o přiblížení práce v programu POV-Ray, o popis pomocného textového souboru, který jsem vytvořil, a o popis některých úloh z Mongeova promítání, které jsem vykreslil. V práci se zaměřuji především na popis tvorby jednotlivých scén v programu POV-Ray a jen okrajově se věnuji způsobu řešení úloh z teorie Mongeova promítání.

Celou práci jsem rozdělil do tří větších kapitol.

V první kapitole popisuji program POV-Ray, práci v něm a základní nastavení scén, které později používám.

Druhá kapitola je věnována pomocnému textovému souboru, který v každé scéně používám a do kterého jsem si předem nadefinoval většinu objektů, které se ve scénách objevují.

Ve třetí, nejobsáhlejší kapitole, popisuji již jednotlivé scény znázorňující úlohy v Mongeově promítání. Nejprve jsem se snažil o zobrazení základních útvarů jako bod, přímka a rovina. Poté jsem vykreslil některé polohové úlohy, dále metrické úlohy a na závěr jsem zobrazil úlohy, ve kterých vystupují tělesa.

Na CD, které je k této práci k přiložene, jsou k dispozici všechny zdrojové kódy, samostatné přeložené obrázky a instalační soubor pro program POV-Ray. Zdrojové kódy jsou v neskrácené podobě a se všemi poznámkami, které jsem si při jejich vytváření psal.

1 Program POV-Ray

1.1 Úvodní informace o programu

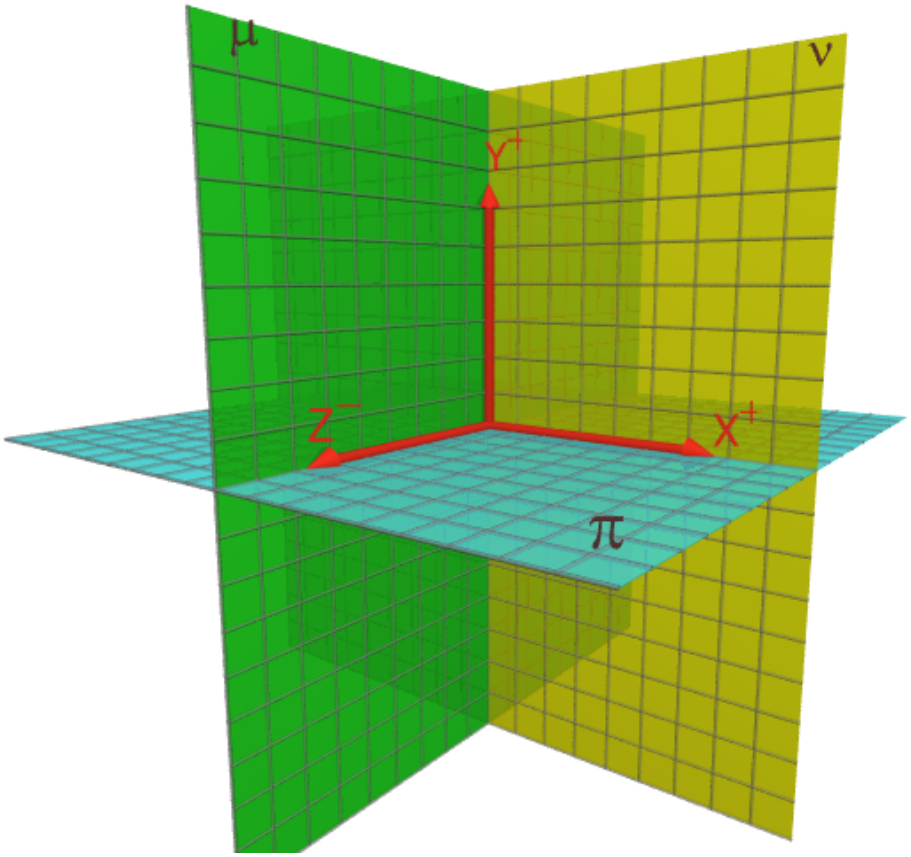
Citováno ze serveru zabývajícího se grafikou:
V roce 1986 uvedl programátor David Buck svůj první Raytracer DKBTrace (David Kirk Buck's Trace) pro Amigu založený na volném zdrojovém kódu v C pro Unix. O něco později, v letech 1987–1988 společně s jiným programátorem Aaronem Collinsem rozšířili DKBTrace i na platformu PC. Poslední verze DKBTrace nesla označení 2.12 (rok 1989).
Program se těšil velké popularitě, ale protože oba programátoři nestačili programovat nové verze dostatečně rychle (alespoň podle mínění věčně nespokojených uživatelů), začalo se mluvit na diskuzních fórech CompuServe o napsání úplně nového raytraceru.
David Buck neváhal a poslal na fórum zprávu, že poskytne DKBTrace jako základ pro tento nový raytracer za splnění třech podmínek. Výsledný program musí být freeware s přístupným zdrojovým kódem (dnes bychom řekli OpenSource), musí být nadále vyvíjen pro různé platformy a musí mít nové jméno. A tak vznikl POV–Ray – Persistence Of Vision Raytracer, jehož poslední verze má k dnešnímu dni pořadové číslo 3.62.

Program POV-Ray je software pro vytváření obrázků 3D počítačové grafiky pomocí metody sledování paprsku, tzv. raytracing. Celá scéna je popsána v textovém souboru jazykem označovaným SDL (*scene description language*). Takto popsanou scénu POV-Ray přečte, provede vykreslení (rendering) a výsledek uloží ve formě rastrového obrázku o předem zadaném rozlišení.

1.2 Nastavení scény

Aby výsledný obraz vypadal přesně podle našich představ, je potřeba v hlavičce zdrojového kódu určit několik parametrů kreslené scény. Ke správnému nastavení

je potřeba vědět, že vše se ve scéně popisuje vzhledem k pevnému systému souřadnic, který je implicitně nastaven jako levotočivý a je znázorněn na následujícím obrázku. V případě potřeby se dá toto nastavení změnit a souřadný systém může být i pravotočivý. Já jsem zůstal u nastaveného levotočivého systému.



Obrázek 1: Levotočivá soustava souřadnic

Při určování polohy objektů je tedy důležité správně zadat souřadnice objektu. Základní kostrou scény je kamera, která promítá scénu na obrazovku, světlo osvětlující objekty scény (může jich být víc) a konečně objekty, které tvoří scénu. Nastavení parametrů kamery a světla jsem pro každý obrázek volil individuálně s ohledem na názornost zobrazované scény. Pro nastavení kamery jsem si nadefinoval parametr `camrot`, který určí natočení kamery ve směru os x a y. Tento příkaz také používám při tvorbě textu popisujícího scénu a díky němu je text vždy natočen směrem ke kameře.

```
#declare camrot = <10, -37, 0>
```

Kameru jsem nejčastěji umisťoval následujícím způsobem:

```
camera {location <0, 0, -12> rotate camrot look_at <0, 0, 0>}
```

Její pozici jsem zvolil na souřadnicích `(0, 0, -12)` a následně jsem ji otočil příkazem `rotate camrot` o zvolené hodnoty. Poslední parametr udává místo, kam se kamera dívá `(0, 0, 0)`.

Světelné zdroje jsem se rozhodl použít dva, jeden vzdálenější umístěný na souřadnicích `(250, 250, -150)` a jeden bližší `(10, 15, -15)`. Pomocí příkazu `shadowless` jsem u nich zakázal tvorbu vržených stínů, které by jednotlivé scény činily méně přehlednými.

```
light_source {<250, 250, -150> color White shadowless}
light_source {<10, 15, -15> color White shadowless}
```

2 Pomocný textový soubor

V souboru

```
pomocne_prvky.pov
```

jsem si předem nadefinoval barvy, přímky, ploch π, ν, μ, pomocné čáry, popisky a samotné objekty, ze kterých se scény skládají.

2.1 Nastavení barev

Barvy jsem volil tak, aby se daly použít pro všechny scény stejné. Uvedu jen několik příkladů.

```
//------------------------------ barvy ----------------------------
#declare Primka = rgb <0.75, 0.15, 0.1> ;                // červená
#declare Prumet = color Navy ;                           // modrá
#declare Stopa = color MidnightBlue filter .3;           // tmavě modrá
```

Stejným způsobem jsem deklaroval i barvy dalších těles, viz přiložené CD.

2.2 Plochy znázorňující průmětny

Pro objekty, ve scénách požívané, jsem použil příkaz umožňující definovat nové funkce a procedury. Je to příkaz:

```
#macro Název(parametr_1, ..., parametr_N)
```

⋮

```
#end.
```

Nové objekty se v tomto prostředí popisují pro obecné parametry a teprve až když chceme objekt vykreslit parametry konkretizujeme. Vše jsem podrobněji popsal na následujícím příkladu tvorby plochy ν.

```
//--------------------------- plocha xy0 ----------------------------
#declare H = 0.01;
#macro Plocha_xy0(Delka)
  box { <-Delka, -Delka, -H/2>, <Delka, Delka, H/2>
          pigment {color barva2p} }
  #declare B = -Delka;
```

```
    #while (B < Delka)
        cylinder { <B, Delka, 0> <B, -Delka, 0> 2*H
                   pigment {color barva_dark} }
        cylinder { <Delka, B, 0> <-Delka, B, 0> 2*H
                   pigment {color barva_dark} }
        #declare B = B + 0.5;
    #end
#end
```

Hodnota `H` udává šířku roviny. Text `Plocha_xy0(Delka)` je odkaz, pomocí kterého se makro v popisu scény volá. Parametr `Delka` udává délku a výšku roviny směrem od počátku. Například `Plocha_xy0(5)` vykreslí ve scéně plochu o délce a výšce 10 jednotek (pět na každou stranu směrem od počátku). Příkazem `box` vykresluji kvádr určený souřadnicemi spodního levého a protějšího horního pravého vrcholu. Pomocí cyklu `while` jsem na rovinu pro lepší představu nanesl ještě čtvercovou síť. Vzdálenost těchto „ordinál“ je na ose x vždy 0,5 jednotky. V příloze jsou popsané ještě zbylé dvě roviny π a μ.

2.3 Znázornění bodu

Body jsem se rozhodl zobrazovat jako koule o malém poloměru.

```
//---------------------------- bod ----------------------------
#macro bod(S1, S2, S3, Poloměr, Barva)
  sphere { <S1, S2, S3>,
           Poloměr, pigment{color Barva} }
#end
```

Při vykreslení bodu použiji, podobně jako u roviny, makro `bod(S1, S2, S3, Poloměr, Barva)`. Parametry `S1, S2, S3` udávají souřadnice bodu. Pro zobrazení koule je potřeba zadat ještě její poloměr a barvu, což jsou zbylé dva parametry v makru.

2.4 Znázornění přímky

Přímky jsem vykresloval jako válce o malém poloměru podstavy.

```
//------------------------- primka -------------------------
#macro primka(P1, P2, P3, K1, K2, K3, Polomer, Barva)
```

```
  cylinder { <P1, P2, P3>, <K1, K2, K3>,
             Polomer, pigment {Barva} }
#end
```

Pro vykreslení válce je potřeba zadat souřadnice středů jeho spodní a horní podstavy `(P1, P2, P3, K1, K2, K3)`, poloměr podstavy `Polomer` a opět barvu tělesa.

2.5 Znázornění pomocných čar

V obrázcích se také často vyskytují pomocné čáry, které nahrazují čárkované přímky při klasickém rýsování. Pro jejich tvorbu jsem použil následující marko. Zvlášť jsem rozlišoval svislé a vodorovné pomocné čáry.

```
//----------------------- pomocna cara svisla ------------------------
#macro Pomocna_cara_svisla(P1, P2, P3, Q1, Q2, Q3, Polomer, Barva)
  #declare A1 = P2;
  #declare B1 = P2+0.3;
     #while (B1 <= Q2)
        cylinder {<P1, A1, P3> <P1, B1, P3> 3*Polomer pigment{Barva} }
        #declare A1 = A1 + 0.6;
        #declare B1 = B1 + 0.6;
     #end
#end

//----------------------- pomocna cara vodorovna ---------------------
#macro Pomocna_cara_vodorovna(R1, R2, R3, S1, S2, S3, Polomer, Barva)
  #declare A2 = R3;
  #declare B2 = R3-0.3;
     #while (B2 >= S3)
        cylinder {<R1, R2, A2> <R1, R2, B2> 3*Polomer pigment{Barva} }
        #declare A2 = A2 - 0.6;
        #declare B2 = B2 - 0.6;
     #end
#end
```

Jak si můžete všimnout, tak pomocí cyklu *while* měním vždy y-ovou (svislá přímka) nebo z-ovou (vodorovná přímka) souřadnici středů podstav válce. A to tak aby vždy mezi jednotlivými válci byla mezera, rovna jejich výšce. Tyto souřadnice měním tak dlouho, dokud hodnota na středu horní podstavy není menší nebo rovna zvolenému parametru.

2.6 Znázornění šipky

Pro znázornění směru otočení roviny při Mongeově promítání jsem si předem nachystal šipku jdoucí směrem otočení.

```
//------------------------- sipka --------------------------
#macro sipka(Polomer, Tloustka, Posun, Barva)
difference {
    torus {Polomer, Tloustka pigment {color Barva}
            rotate z*90
            translate <Posun, 0, 0>}
    box{ <Posun - 0.2, -2*Tloustka, - 2*Polomer + 2*Tloustka>
         <Posun + 0.2, 2*Polomer + 2*Tloustka, 2*Polomer + 2*Tloustka>}
    box{ <Posun - 0.2, - 2*Polomer + 2*Tloustka, -0.5>
         <Posun + 0.2, 2*Tloustka, 2*Polomer + 2*Tloustka>} }
cone {<Posun, -Polomer + 0.05, -0.5>, 3*Tloustka
      <Posun, -Polomer, -0.1>, 0 pigment {color Barva}}
#end
```

Pro její tvorbu jsem použil dvě tělesa. Základem šipky je anuloid (v textovém souboru zvaný `torus`), k němu jsem musel udělat dva kvádry a z těchto těles jsem poté pomocí příkazu pro rozdíl vytvořil oblouk šipky. Parametry šipky jsou první a druhý poloměr anuloidu, souřadnice udávající jeho posun na ose x a barva anuloidu. Aby výsledné těleso opravdu připomínalo šipku, umístil jsem ještě na spodní okraj vzniklého oblouku kužel určující směr šipky.

2.7 Znázornění roviny v obecné poloze

V obrázcích, zaměřujících se na zobrazení roviny, jsem používal následující makro.

```
//------------------------- rovina -------------------------
#macro rovina(Px1, Py1, Pz1, Px2, Py2, Pz2, Px3, Py3, Pz3,
              Z_souradnice_D, Barva)
#declare ur1 = Px2-Px1 ;                //vektor B-A
#declare ur2 = Py2-Py1 ;                //vektor B-A
#declare ur3 = Pz2-Pz1 ;                //vektor B-A
#declare vr1 = Px3-Px1 ;                //vektor C-A
#declare vr2 = Py3-Py1 ;                //vektor C-A
#declare vr3 = Pz3-Pz1 ;                //vektor C-A
#declare vs1 = ur2*vr3-ur3*vr2 ;        //vektorový součin
#declare vs2 = ur3*vr1-ur1*vr3 ;        //vektorový součin
```

```
#declare vs3 = ur1*vr2-ur2*vr1 ;         //vektorový součin
#declare d = -vs1*Px1-vs2*Py1-vs3*Pz1 ;  // konst. d v rovnici roviny
#declare X_P = -d/vs1 ;                  // prusecik s osou x
//------------------------- souradnice bod D ----------------
#declare Pz4 = Z_souradnice_D ;
#declare Px4 = (-d-vs3*Pz4)/vs1 ;
#declare Py4 = 0 ;
//------------------------- rovina --------------------------
polygon {5, <Px1, Py1, Pz1>
            <Px2, Py2, Pz2>
            <Px3, Py3, Pz3>
            <Px4, Py4, Pz4>
            <Px1, Py1, Pz1>
            pigment{Barva filter .8} }
#end
```

Rovinu jsem vykresloval pomocí příkazu `rovina(Px1, Py1, Pz1, Px2, Py2, Pz2, Px3, Py3, Pz3, Z_souradnice_D, Barva)`. Parametry představují souřadnice tří různých bodů určujících rovinu, z-ovou souřadnici čtvrtého bodu D a barvu roviny. Jelikož jsem nechtěl libovolnou rovinu vykreslovat jen jako „trojúhelník“ potřeboval jsem ze zadaných tří bodů zjistit i další (čtvrtý) bod roviny. Pomocí znalostí z analytické geometrie jsem ze tří zvolených bodů určil normálový vektor dané roviny. Zvolil jsem si z-ovou souřadnici bodu D a zbylé souřadnice následně dopočítal. Samotnou rovinu jsem na závěr vykreslil pomocí příkazu pro tvorbu mnohoúhelníků `polygon{počet vrcholů+1, jejich souřadnice, barva}`

2.8 Znázornění hlavních přímek roviny

S rovinami v Mongeově promítání úzce souvisí hlavní přímky roviny. Pro jejich tvorbu jsem si opět napsal krátký skript, pomocí kterého je ve scéně vykresluji.

```
//------------------------- hlavni primky 1 osnovy ------------------
#macro hl_primka_1(Px1, Py1, Pz1, Px2, Py2, Pz2, Px3, Py3, Pz3,
                   Z_roviny_Ro2, Y_roviny_Ro2, Polomer, Barva)
#declare u1 = Px2-Px1 ;                  // vektor B-A
#declare u2 = Py2-Py1 ;                  // vektor B-A
#declare u3 = Pz2-Pz1 ;                  // vektor B-A
#declare v1 = Px3-Px1 ;                  // vektor C-A
#declare v2 = Py3-Py1 ;                  // vektor C-A
#declare v3 = Pz3-Pz1 ;                  // vektor C-A
```

```
#declare vs1 = u2*v3-u3*v2 ;             // vektorový součin 1
#declare vs2 = u3*v1-u1*v3 ;             // vektorový součin 2
#declare vs3 = u1*v2-u2*v1 ;             // vektorový součin 3
#declare d = -vs1*Px1-vs2*Py1-vs3*Pz1 ;  // konst d v rovnici roviny
#declare Yp1 = Y_roviny_Ro2 ;            // y souradnice rovina rezu
#declare Zp1 = Z_roviny_Ro2 ;            // voleny bod
#declare Xp1 = ((-d-vs2*Yp1)/vs1)-(vs3*Zp1)/vs1 ;
#declare Yp2 = Y_roviny_Ro2 ;
#declare Zp2 = 0 ;
#declare Xp2 = ((-d-vs2*Yp2)/vs1)-(vs3*Zp2)/vs1 ;
cylinder { <Xp1, Yp1, Zp1> <Xp2, Yp2, Zp2>
           Polomer   pigment {Barva} }
#end
```

Jako parametry spouštějícího příkazu pro tvorbu hlavních přímek první osnovy jsem zvolil opět tři body určující rovinu, z-ovou souřadnici krajního bodu vykreslované hlavní přímky, y-ovou souřadnici roviny řezu, poloměr podstavy válce znázorňujícího hlavní přímku a barvu této přímky. Zmíněná rovina řezu je rovina σ rovnoběžná s povinou π a jejíž průsečík se zadanou rovinou α je právě hledaná hlavní přímka. Odvozené souřadnice `Xp1, Yp1, Zp1, Xp2, Yp2, Zp2` koncových bodů úsečky, znázorňující hlavní přímku, jsem získal z analytického vyjádření průniku rovin α a σ.

Analogickým způsobem tvořím i hlavní přímky druhé osnovy a příkaz pro jejich tvorbu je také k dispozici na přiloženém CD.

2.9 Znázornění přímky kolmé k rovině

Pro znázornění kolmice ke zvolené rovině jsem vytvořil následující makro.

```
//------------------------- kolmice k rovine -------------------
#macro kolma_primka(Px1, Py1, Pz1, Px2, Py2, Pz2, Px3, Py3, Pz3,
                    bod_1, bod_2, bod_3, Parametr, Polomer, Barva)
#declare ur1 = Px2-Px1 ;             //vektor B-A
#declare ur2 = Py2-Py1 ;             //vektor B-A
#declare ur3 = Pz2-Pz1 ;             //vektor B-A
#declare vr1 = Px3-Px1 ;             //vektor C-A
#declare vr2 = Py3-Py1 ;             //vektor C-A
#declare vr3 = Pz3-Pz1 ;             //vektor C-A
#declare vs1 = ur2*vr3-ur3*vr2 ;     //vektorový součin
#declare vs2 = ur3*vr1-ur1*vr3 ;     //vektorový součin
#declare vs3 = ur1*vr2-ur2*vr1 ;     //vektorový součin
```

```
//------------------------- spodni bod primky -------------------
#declare Pp1 = bod_1 ;
#declare Pp2 = bod_2 ;
#declare Pp3 = bod_3 ;
//------------------------- horni bod primky --------------------
#declare S = Parametr ;              // parametr posunu horniho bodu
#declare Ph1 = Pp1 + S*vs1 ;
#declare Ph2 = Pp2 + S*vs2;
#declare Ph3 = Pp3 + S*vs3;
//------------------------- primka ------------------------------
cylinder { <Pp1, Pp2, Pp3>,
           <Ph1, Ph2, Ph3>,
           Polomer pigment {Barva} }
#end
```

V tomto případě je parametrů ve volající funkci více. Jsou to opět souřadnice tří bodů určujících rovinu, ke které chceme kolmici konstruovat. Potom souřadnice krajního bodu kolmice ležícího v rovině π, parametr posunu, pomocí kterého zjišťuji druhý krajní bod kolmice, a na závěr průměr a barva kreslené kolmice. Při tvorbě kolmice opět vycházím ze znalostí z analytické geometrie, konkrétně toho, že normálový vektor roviny je směrovým vektorem hledané kolmice. A dále z parametrického vyjádření přímky, kde pomocí zvoleného parametru posunu a spočítaného směrového vektoru najdu souřadnice druhého krajního bodu kolmice.

2.10 Znázornění značky pro pravý úhel

S kolmicemi úzce souvisí další objekt, který jsem si dopředu nadefinoval. Je to znak pro pravý úhel.

```
//------------------------- znak praveho uhlu -------------------------
#macro pravy_uhel(Polomer, Tloustka, Barva)
union {
   intersection  {
           torus {Polomer, Tloustka pigment {color Barva} }
             box {<0 ,0 - 0.15, 0>,
                  <Polomer + 0.15, 0 + 0.15, -Polomer - 0.15>
                  pigment {color Barva} }
                 }
   sphere {<Polomer/2.8, 0, -Polomer/2.8>
           Tloustka + 0.01 pigment {color Barva} }
}
#end
```

Parametry jsou poloměr torusu, tvořící oblouk u této značky, tloušťka tohoto torusu a jeho barva. Oblouk jsem vytvořil pomocí průniku vhodného kvádru a zmíněného torusu. A poté co jsem oblouk vytvořil jsem k němu ještě připojil bod o vhodných souřadnicích. Ten znázorňuje tečku ve značce pro pravý úhel.

2.11 Znázornění průsečíku přímek

Při tvorbě několika scén jsem potřeboval znát souřadnice průsečíku přímek, které byly zadány dvěma svými různými body. Celou úlohu jsem rozdělil na tři případy, podle toho kde se přímky, jejichž průsečík hledám, nachází. Průsečík přímek v nárysně, v půdorysně a obecně v prostoru. Rozdělil jsem to tak z důvodu lepší práce se získanými souřadnicemi hledaného bodu. Pro obsáhlost celého textového souboru uvedu jen ukázku průsečíku přímek v nárysně, zbylé případy jsou analogické a jsou opět k dispozici na přiloženém CD.

```
//-------------------------- prusecik primek --------------------------
#macro prusecik_primek_nar(Px1, Py1, Pz1, Px2, Py2, Pz2,
                  Px3, Py3, Pz3, Px4, Py4, Pz4, Polomer, Barva)
#declare un1 = Px2-Px1 ;              //vektor C-A
#declare un2 = Py2-Py1 ;              //vektor C-A
#declare un3 = Pz2-Pz1 ;              //vektor C-A
#declare vn1 = Px4-Px3 ;              //vektor G-E
#declare vn2 = Py4-Py3 ;              //vektor G-E
#declare vn3 = Pz4-Pz3 ;              //vektor G-E
#declare s = (un1*(Py1-Py3)+un2*(Px3-Px1))/(un1*vn2-vn1*un2) ;
#declare B1 =  Px3 + s*vn1 ;
#declare B2 =  Py3 + s*vn2 ;
#declare B3 =  Pz3 + s*vn3 ;
sphere { <B1, B2, B3>, Polomer
        pigment{color Barva} }
#end
```

Parametry jsou krajní body úseček znázorňujících přímky jejichž průsečík hledáme, poloměr koule, která bude průsečík znázorňovat a barva této koule. Tak jako v předchozích příkladech si spočítám směrové vektory přímek, pomocí nich vyjádřím z parametrických rovnic přímek parametr s a díky němu zjistím souřadnice B_1, B_2, B_3 hledaného průsečíku.

2.12 Znázornění pomocné plochy

Další objekt, který jsem si předem nadefinoval jsem nazval pomocnou plochou a jedná se o čtyřúhelník usnadňující představu o kolmém průmětu přímky do nárysny a půdorysny.

```
//-----------------------pomocne plochy-----------------------
#macro pomocne_plochy(PX1, PY1, PZ1,
                      PX2, PY2, PZ2)
   polygon {5, <PX1, PY1, PZ1>,
               <PX2, PY2, PZ2>,
               <PX2, PY2, 0>,
               <PX1, PY1, 0>,
               <PX1, PY1, PZ1>
               pigment {barva_rovina_svisla} }
   polygon {5, <PX1, PY1, PZ1>,
               <PX2, PY2, PZ2>,
               <PX2, 0, PZ2>,
               <PX1, 0, PZ1>,
               <PX1, PY1, PZ1>
               pigment {barva_rovina_vodorovna} }
#end
```

Parametry jsou jen krajní body úsečky, představující přímku, ze které chceme čtyřúhelníky konstruovat. Poté jsem využil příkazu pro tvorbu mnohoúhelníků `polygon` a vhodně upravil jednotlivé souřadnice bodů které tvoří vrcholy kresleného čtyřúhelníku.

2.13 Znázornění popisu objektů

Důležité objekty jsem ve scénách popisoval textem, pro jehož tvorbu jsem vytvořil následující marka. Rozlišoval jsem dva typy popisků. První, který je určen pro popisování rovin a druhý kterým popisuji všechny zbylé objekty. Udělal jsem to z důvodu snadnějšího zadávání příkazů pro tvorbu tohoto textu a hlavně z toho důvodu, že roviny je zvykem popisovat řeckými písmeny, zatímco objekty jako body nebo přímky se popisují písmeny latinské abecedy.

```
//------------------------- popisky --------------------------
#macro pismeno(Znak, Horni_Index_levy, Horni_Index_pravy, Dolni_Index,
               Souradnice_X, Souradnice_Y, Souradnice_Z, Barva)
union{  text { ttf "tahoma", Znak, H, 0
```

```
                pigment {Barva}
                        scale 0.5 }
          text { ttf "tahoma", Horni_Index_levy, H, 0
                pigment {Barva}
                        scale 0.3
                        translate <-0.3,0.26,0> }
          text { ttf "symbol", Horni_Index_pravy, H, 0
                pigment {Barva}
                        scale 0.35
                        translate <0.3,0.26,0> }
          text { ttf "tahoma", Dolni_Index, H, 0
                pigment {Barva}
                        scale 0.3
                        translate <0.3,-0.1,0> }
        rotate camrot
        translate <Souradnice_X, Souradnice_Y, Souradnice_Z> }
#end

#macro popis_roviny(Znak_p, Souradnice_Xp, Souradnice_Yp,
                    Souradnice_Zp, Barva_p)
text { ttf "symbol", Znak_p, H, 0
        pigment {Barva_p}
                 scale 0.5
       rotate camrot
       translate <Souradnice_Xp, Souradnice_Yp, Souradnice_Zp> }
#end
```

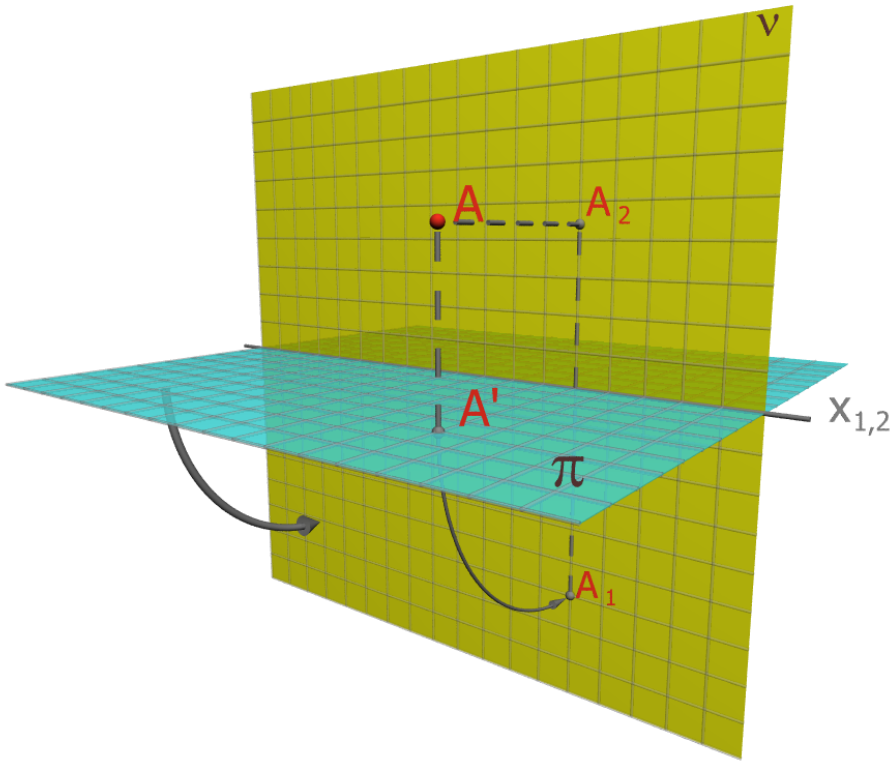
U textu popisujícího jiné objekty než roviny jsou kromě souřadnic a barvy textu ještě čtyři parametry. První tvoří hlavní text, který se má vykreslit a další tři parametry jsou jeho indexy. Jsou to po řadě horní levý, horní pravý a dolní pravý index. U horního pravé indexu používám styl textu `symbol` a díky tomu je index vykreslován jako písmeno řecké abecedy. Jak si můžete všimnout, tak je zde použit parametr `camrot`, který je popsán a nadefinován v kapitole 1.2, při popisu nastavení kamery. Tento parametr zajistí natočení textu směrem ke kameře, díky čemuž by měl být text čitelnější.

3 Mongeovo promítání modelované pomocí programu POV-Ray

3.1 Zobrazení základních útvarů

3.1.1 Zobrazení bodu

Mongeovo promítání je pravoúhlé promítání na dvě k sobě kolmé průmětny. Jedna se obvykle značí π a nazývá se půdorysna, druhá se značí ν a nazývá se nárysna. Bod v Mongeově promítání nejprve pravoúhle promítneme do půdorysny π, jako bod A', a do nárysny ν, jako bod A_2. Poté sklopíme kolem osy x půdorysnu do nárysny. Tím každému dobu v prostoru jednoznačně přiřadíme dvojici bodů v rovině, ležících na jedné přímce (ordinále) kolmé k ose x. Konkrétně bod A' sklopíme do bodu A_1.



Obrázek 2: Zobrazení bodu

Textový soubor, pomocí kterého program POV-Ray scénu vykreslí, se pokusím popsat podrobně.

```
//------------------------------ bod --------------------------
#include "colors.inc"
#include "pomocne_prvky.pov"
#declare camrot = <10,-37,0>;
camera { location <0, 0, -12> rotate camrot look_at <0, 0, 0>}
light_source{<250,250,-150> color White shadowless}
light_source{<10,15,-15> color White shadowless}
//------------------------ plochy -----------------------------
Plocha_xy0(4.5)
Plocha_x0z(4.5)
primka(-5, 0, 0, 5, 0, 0, 3*H, barva_dark)        //osa x_1,2
//--------------------- souradnice bod A ----------------------
#declare Px1 = 2 ;
#declare Py1 = 2.2 ;
#declare Pz1 = -3 ;
//------------------------------ bod A -------------------------
bod(Px1, Py1, Pz1, 8*H, Primka)                  // bod A
bod(Px1, 0, Pz1, 7*H, Pomoc)  // bod A'
bod(Px1, Py1, 0, 7*H, Pomoc)        // bod A2
bod(Px1, Pz1, 0, 7*H, Pomoc)  // bod A1
//-------------------------- pomocne cary ----------------------
Pomocna_cara_vodorovna(Px1, Py1, 0, Px1, Py1, Pz1, 0.01, Pomoc)
Pomocna_cara_svisla(Px1, 0, Pz1, Px1, Py1, Pz1, 0.01, Pomoc)
Pomocna_cara_svisla(Px1, Pz1, 0, Px1, Py1, 0, 0.01 Pomoc)
//--------------------------- sipky ----------------------------
sipka(3, 0.06, -3, Pomoc)
sipka(-Pz1, 0.02, Px1, Pomoc)
//------------------------------ popisky -----------------------
pismeno("A"," "," ", Px1 + 0.2, Py1, Pz1, Primka)
pismeno("A'","","  ", Px1 + 0.5, 0.2, Pz1 - 0.3, Primka)
pismeno("A"," ","2", Px1 + 0.3, Py1 + 0.1, - 0.3, Primka)
pismeno("A"," ","1", Px1 + 0.3, Pz1 + 0.2, - 0.3, Primka)
popis_roviny("p", 4.2, 0.2, -4.4 , barva_popis_roviny)
popis_roviny("n", 4.2, 4.2, -0.2 , barva_popis_roviny)
pismeno("x"," ","1,2", 5.2, 0, 0, barva_dark)
```

Nejprve jsem načetl pomocné knihovny. Knihovnu nazvanou `colors.inc` obsahující barvy nadefinované implicitně v programu POV-Ray a mnou vytvořený soubor `pomocne_prvky.pov` popsaný v první kapitole. Ve scéně tedy vykresluji plochu π příkazem `Plocha_x0z(4.5)`, plochu ν příkazem `Plocha_xy0(4.5)` a osu x pomocí příkazu `primka(-5, 0, 0, 5, 0, 0, 3*H, barva_dark)`. Jak si

můžete všimnout, tak za parametr udávající „tloušťku“ osy jsem zvolil trojnásobek šířky rovin `3*H`. Abych nemusel neustále psát číselné hodnoty souřadnic bodů, které chci kreslit, nadefinoval jsem si předem jejich hodnoty a dále jsem se na ně již jen odkazoval.

```
//---------------------- souradnice bod A -----------------------
#declare Px1 = 2 ;
#declare Py1 = 2.2 ;
#declare Pz1 = -3 ;
```

Další makra jsem použil i pro tvorbu zbytku scény. Pomocí makra pro tvorbu bodu jsem vykreslil bod A a jeho příslušné průměty. U tvorby průmětů stačilo vhodně změnit zvolené souřadnice. Dále jsem vykreslil několik pomocných čar, ukazujících promítnutí bodu do příslušné průmětny. Například pomocná čára spojující bod A a bod A_2 je napsána takto:

```
//-------------------------- pomocne cary ----------------------
Pomocna_cara_vodorovna(Px1, Py1, 0, Px1, Py1, Pz1, 0.01, Pomoc)
```

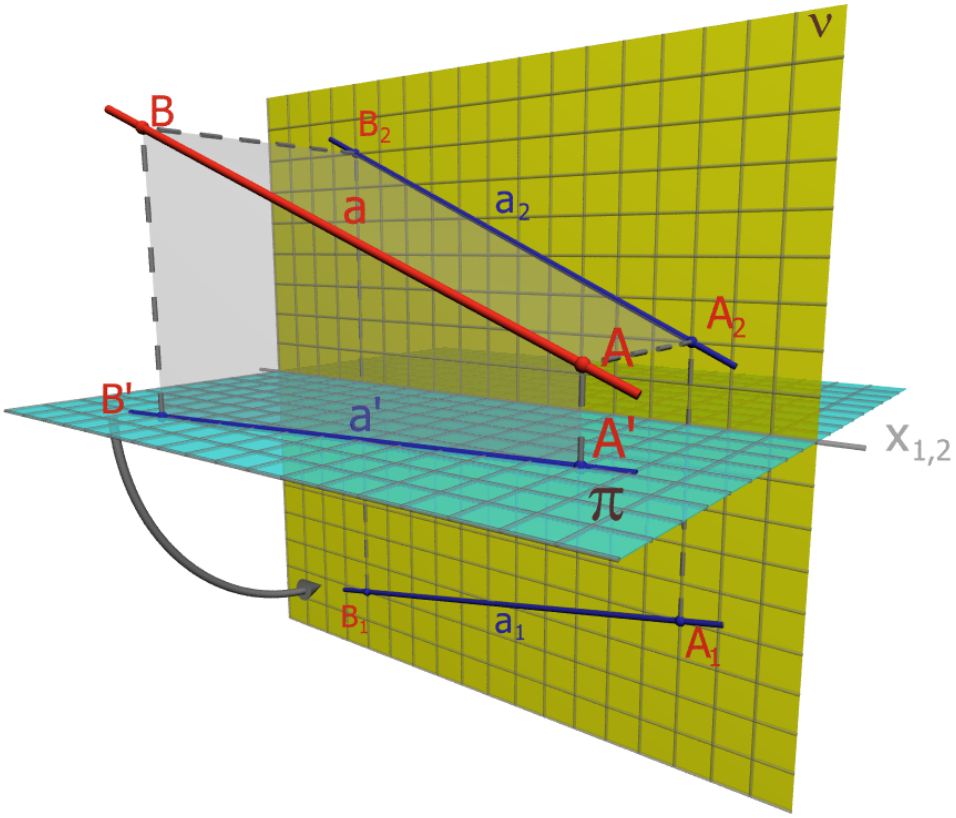
Pro zlepšení představy, jak vzniká bod A_1 jsem do scény přidal ještě dvě šipky naznačující směr sklopení půdorysny do nárysny.

```
sipka(3, 0.06, -3, Pomoc)
```

Na závěr jsem scénu doplnil popisky označujícími jednotlivé objekty ve scéně vykreslované.

3.1.2 Zobrazení přímky

Přímka je v prostoru jednoznačně určena dvěma různými body. Známe-li tyto body, můžeme přímku zobrazit pomocí průmětů jednotlivých bodů, jejichž konstrukci jsem poslal v předchozí kapitole 3.1.1.



Obrázek 3: Zobrazení přímky v obecné poloze určená dvěma body

Jak v tomto případě, tak i v následujících příkladech nebudu podrobně popisovat celý zdrojový text, který scénu popisuje. Uvedu jen některé prvky, kterými se scéna odlišuje od předchozích obrázků. Celý textový soubor je k dispozici na přiloženém CD.

Nejprve jsem zvolil souřadnice bodů A a B které ve zdrojovém kódu značím `Px1, Py1, Pz1` a `Px2, Py2, Pz2`. Snažil jsem se docílit toho, aby se úsečka, znázorňující přímku, nevykreslovala jen od jednoho bodu k druhému, ale aby i pokračovala směrem dál od těchto bodů. Musel jsem proto dopočítat souřadnice

směrového vektoru této přímky a pomocí nich jsem následně získal souřadnice nových dvou bodů ležících na této přímce. Tyto body jsem zvolil jako krajní body úsečky znázorňující kreslenou přímku.

```
//------------------------ prodloužení přímky -----------------------
#declare smer_vektor_x = Px2-Px1 ;
#declare smer_vektor_y = Py2-Py1 ;
#declare smer_vektor_z = Pz2-Pz1 ;
#declare prodluz1 = 0.1 ;               // zvolený parametr prodloužení
#declare prodluz2 = 1.1 ;               // zvolený parametr prodloužení
#declare prodl_x_kl = Px1 + prodluz2 * smer_vektor_x ;
#declare prodl_y_kl = Py1 + prodluz2 * smer_vektor_y ;
#declare prodl_z_kl = Pz1 + prodluz2 * smer_vektor_z ;
#declare prodl_x_za = Px1 - prodluz1 * smer_vektor_x ;
#declare prodl_y_za = Py1 - prodluz1 * smer_vektor_y ;
#declare prodl_z_za = Pz1 - prodluz1 * smer_vektor_z ;
```

Přímky tedy vykresluji příkazy následujícího typu.

```
primka(prodl_x_za, prodl_y_za, prodl_z_za,
       prodl_x_kl, prodl_y_kl, prodl_z_kl, 5*H, Primka)
```

Průměty přímky jsem vykreslil jen vhodnou úpravou souřadnic. Například:

```
//------------------ prumety primky -------------------------
primka(prodl_x_za, prodl_y_za, 0,                   // A2 B2
       prodl_x_kl, prodl_y_kl, 0, 4*H, Prumet)
```

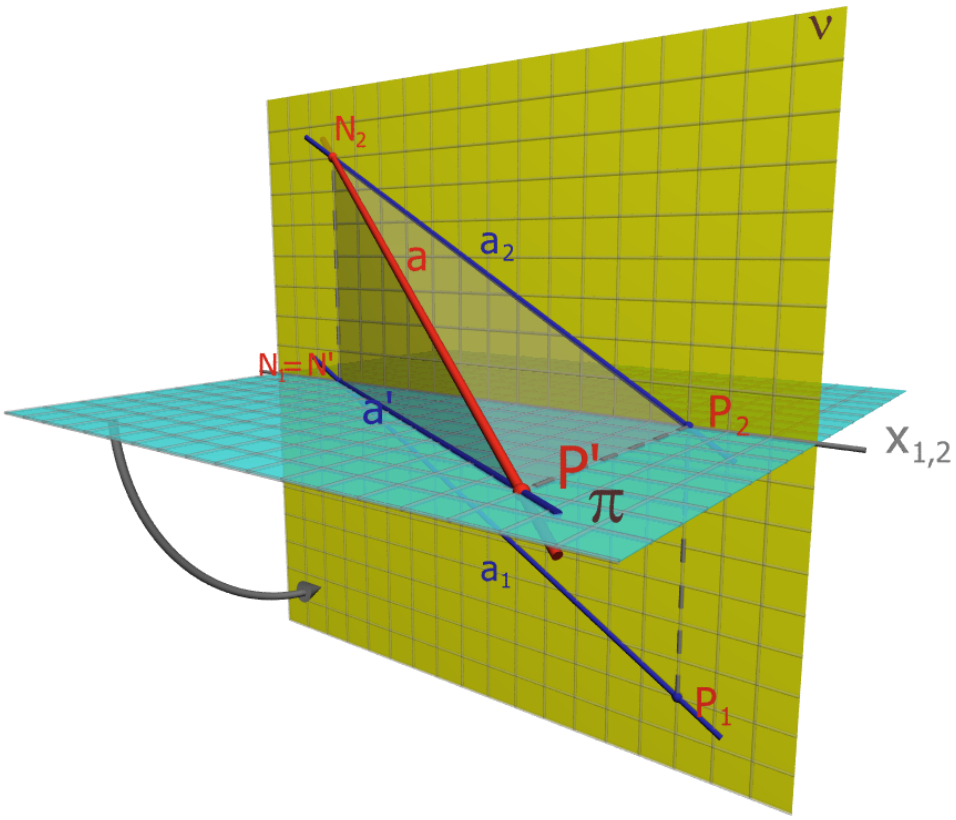
Pomocné čáry jsem nevykresloval z krajních bodů úsečky, ale přímo ze zadaných bodů A, B.

```
Pomocna_cara_svisla(Px2, 0, Pz2, Px2, Py2, Pz2, 0.01 Pomoc)
```

Nově se ve scéně objevují pomocné plochy, ukazující, jakým způsobem se přímka promítá do jednotlivých průměten. Díky vytvořenému makru, které jsem popsal v kapitole 2.12, k tomu stačil následující příkaz.

```
pomocne_plochy(Px1, Py1, Pz1, Px2, Py2, Pz2)
```

Neznáme-li dva různé body přímky můžeme pro její sestrojení využít průsečíky s průmětnami, tzv. stopníky přímky.

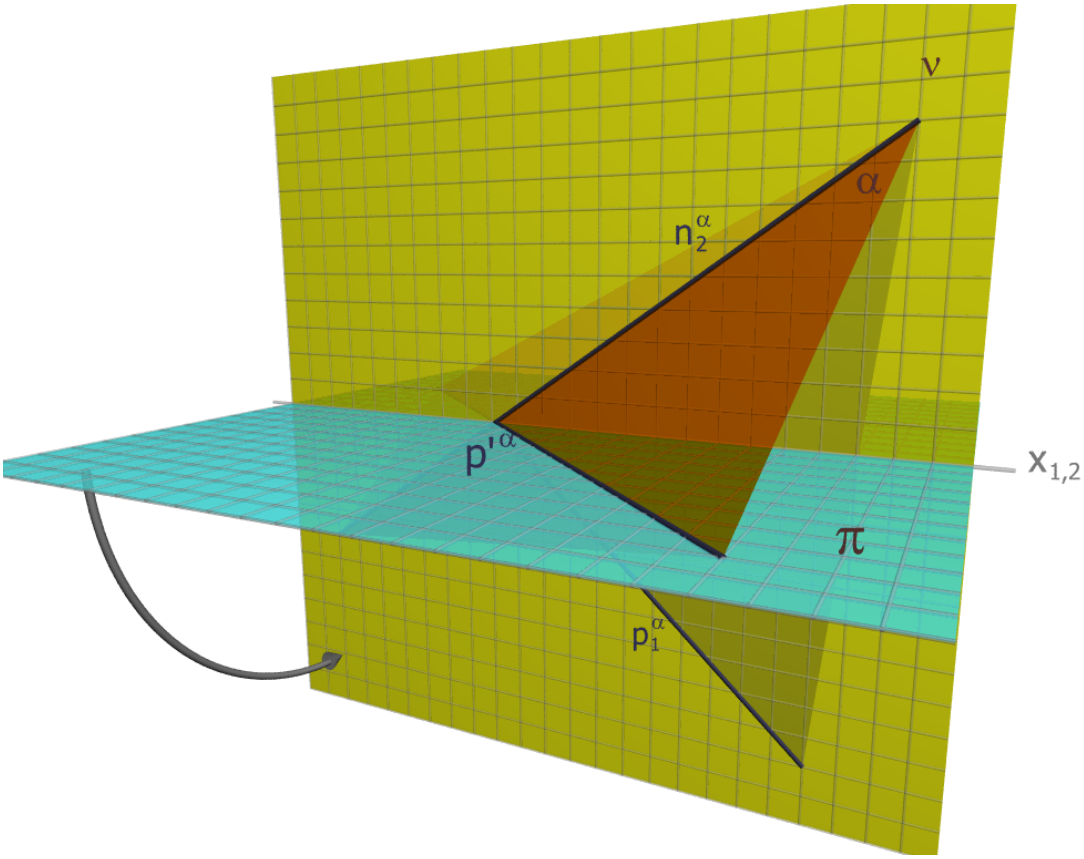


Obrázek 4: Zobrazení přímky v obecné poloze

Zdrojový kód této scény se od kódu předchozího obrázku téměř neliší. Místo bodů A a B jsem zadal přímo souřadnice stopníků P a N. Opět jsem počítal směrový vektor a poté souřadnice bodů přímky, které určí konečnou podobu úsečky znázorňující přímku. Obdobně jako v předchozí úloze jsem pro lepší názornost promítnutí přímky do průměten použil pomocné plochy.

3.1.3 Zobrazení roviny

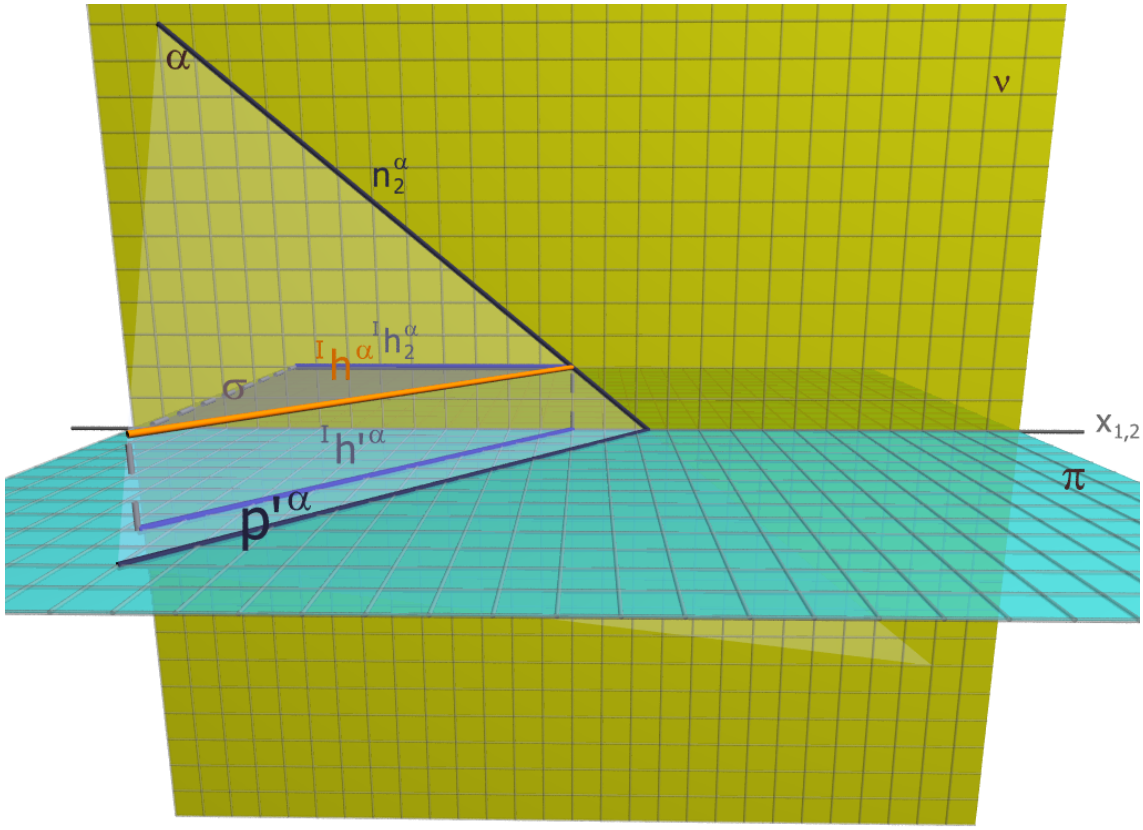
Rovinu lze v prostoru jednoznačně určit několika způsoby. Jak jsem již psal v kapitole 2.7 budu rovinu určovat třemi navzájem různými nekolineárními body. Přesný postup vykreslení roviny pomocí makra, které jsem za tímto účelem vytvořil, jsem také popsal v kapitole 2.7. Na obrázku je rovina znázorněna červeným „obdélníkem“, jehož vrcholy leží v průmětnách. Při popisování roviny jsem se držel zavedeného značení a stopy roviny jsem popisoval obvyklým způsobem. Jak si můžete všimnout na obrázku, tak kromě roviny samotné, jsem se rozhodl zvýraznit ještě její průmět. Na obrázku je vykreslen jako šedý trojúhelník v nárysně.



Obrázek 5: Zobrazení roviny

Při konstrukci mnoha úloh v Mongeově promítání se užívá tzv. hlavních přímek roviny. Jsou to přímky, které vzniknou jako průsečík dané roviny s rovi-

nou rovnoběžnou s půdorysnou nebo nárysnou. Nazývají se hlavní přímky první osnovy a hlavní přímky druhé osnovy.



Obrázek 6: Zobrazení hlavních přímek první osnovy

Při tvorbě této scény jsem k vytvořené scéně pro zobrazení roviny přidal ještě makro pro tvorbu hlavní přímky první osnovy, podrobněji popsaného v kapitole 2.8.

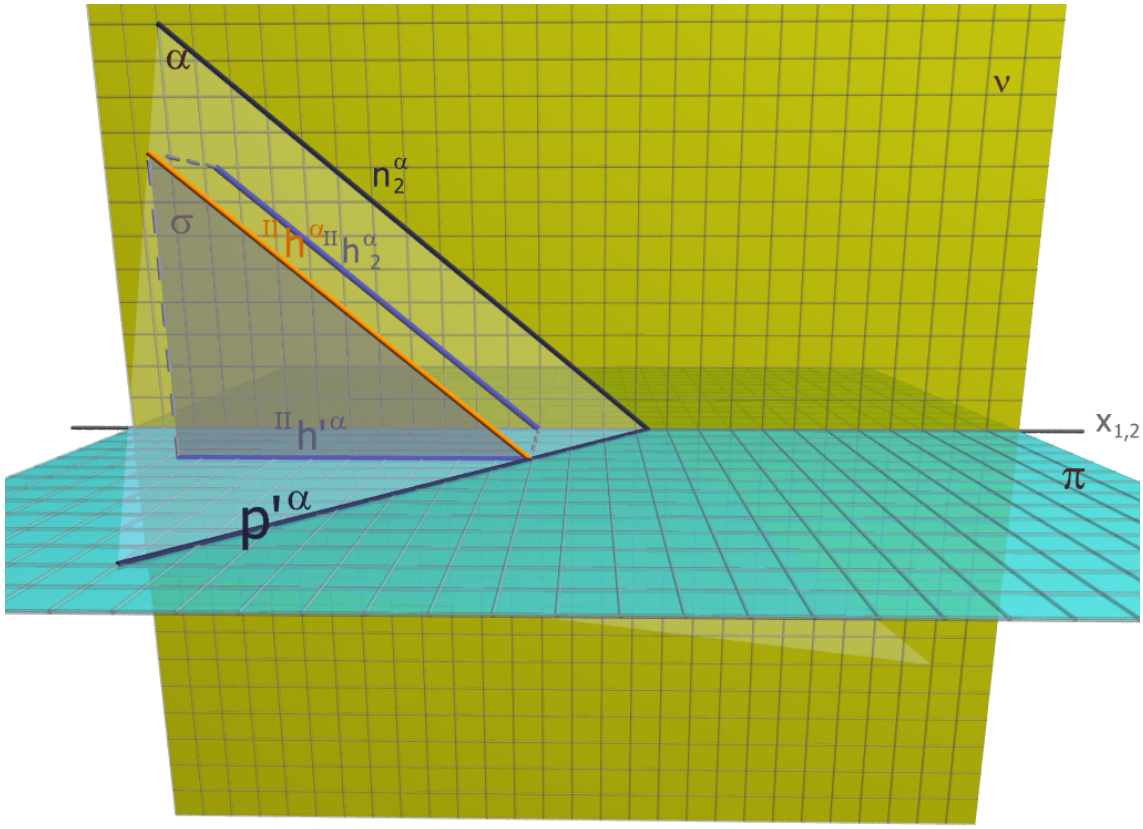
```
//-------------------------- hlavni primka 1 -------------------------
hl_primka_1(Px1, Py1, Pz1, Px2, Py2, Pz2,
            Px3, Py3, Pz3, -5, 1, 4*H, Hl_Primka)
```

Dále jsem vykreslil průměty této přímky a pomocné čáry k této přímce. Pro lepší představu jsem se rozhodl ještě znázornit rovinu σ jako rovinu rovnoběžnou s nárysnou, jejíž průnik s rovinou α je hledaná hlavní přímka.

```
//-------------------------- rovina sigma ----------------------------
polygon{ 4, <Xp1, Yp1, Zp1>
            <Xp1, Yp1, 0>
            <Xp2, Yp2, Zp2>
            <Xp1, Yp1, Zp1>
            pigment {Gray45 filter .9} }
```

Téměř stejným způsobem, jen změnou příslušných souřadnic, jsem vykreslil i scénu s hlavními přímkami druhé osnovy. Pro tvorbu hlavní přímky zde jsem ale využil makro:

```
//-------------------------- hlavni primka 2 -------------------------
hl_primka_2(Px1, Py1, Pz1, Px2, Py2, Pz2,
            Px3, Py3, Pz3, 4, -2, 4*H, Hl_Primka)
```

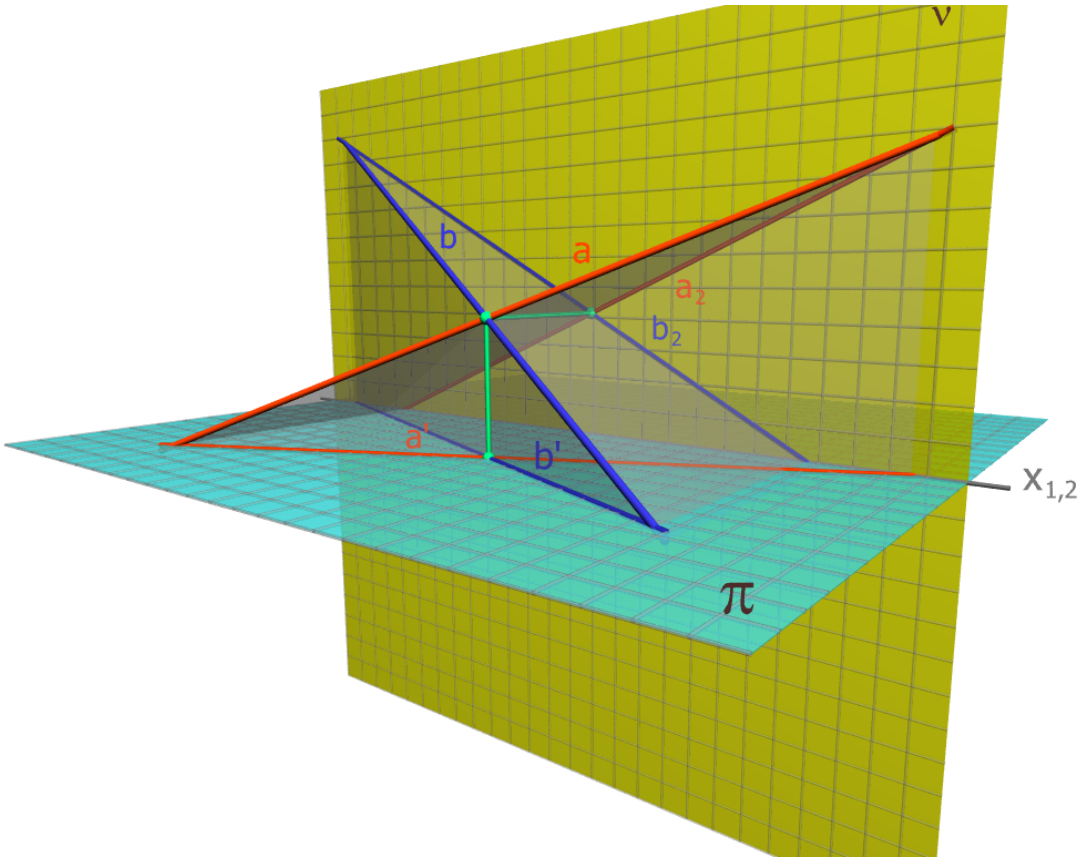


Obrázek 7: Zobrazení hlavních přímek druhé osnovy

3.2 Polohové úlohy

3.2.1 Dvojice přímek

Dvě přímky v prostoru mohou být totožné, rovnoběžné, různoběžné a mimoběžné. Ve své práci jsem se rozhodl ukázat příklady posledních dvou typů vzájemných poloh přímek, tedy různoběžné přímky a mimoběžné přímky.

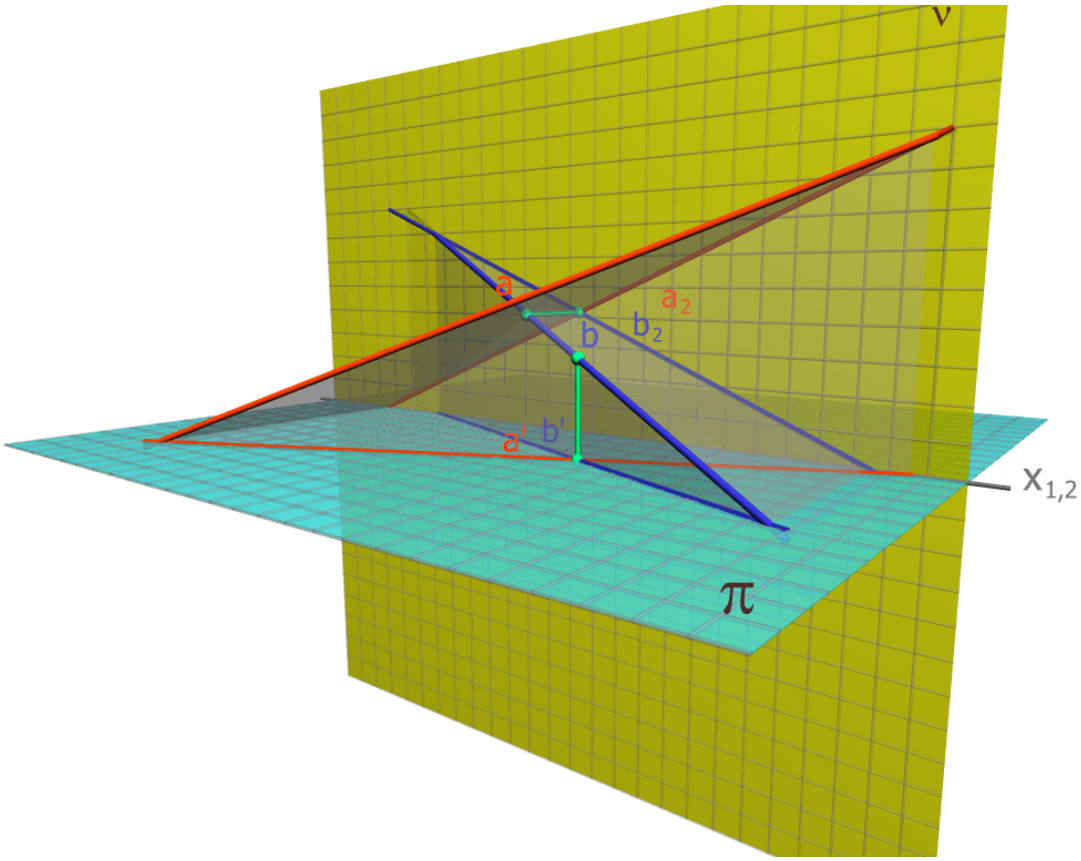


Obrázek 8: Zobrazení různoběžných přímek

V obou scénách jsem přímky určil souřadnicemi dvěma jejich různých bodů. Použil jsem makra pro tvorbu pomocných ploch a pro výpočet souřadnic průsečíků těchto přímek a jejich průmětů. Tyto průsečíky (na nárysně, půdorysně i v prostoru) jsem znázornil body zelené barvy. Z průsečíků v průmětnách jsem poté ještě vedl kolmice k přímkám v prostoru, aby se lépe znázornila jejich vzájemná poloha. Pokud se tyto kolmice protínají v jednom bodě (skutečném průsečíku), jsou dané přímky různoběžné.

```
//------------------- prusecik primek ----------------------------
prusecik_primek_nar(Px1, Py1, 0, Px2, Py2, 0, Px3, Py3, 0,
                    Px4, Py4, 0, 0.1, SpringGreen)        // a2_b2
prusecik_primek_pud(Px1, 0, Pz1, Px2, 0, Pz2, Px3, 0, Pz3, Px4,
                    0, Pz4, 0.1, SpringGreen)             // a´_b´
prusecik_primek_3D(Px1, Py1, Pz1, Px2, Py2, Pz2, Px3, Py3, Pz3,
                   Px4, Py4, Pz4, 0.1, SpringGreen)       // a_b
primka(B1, B2, B3,
       D1, D2, D3, 4*H, SpringGreen)
primka(C1, C2, C3,
       D1, D2, D3, 4*H, SpringGreen)
```

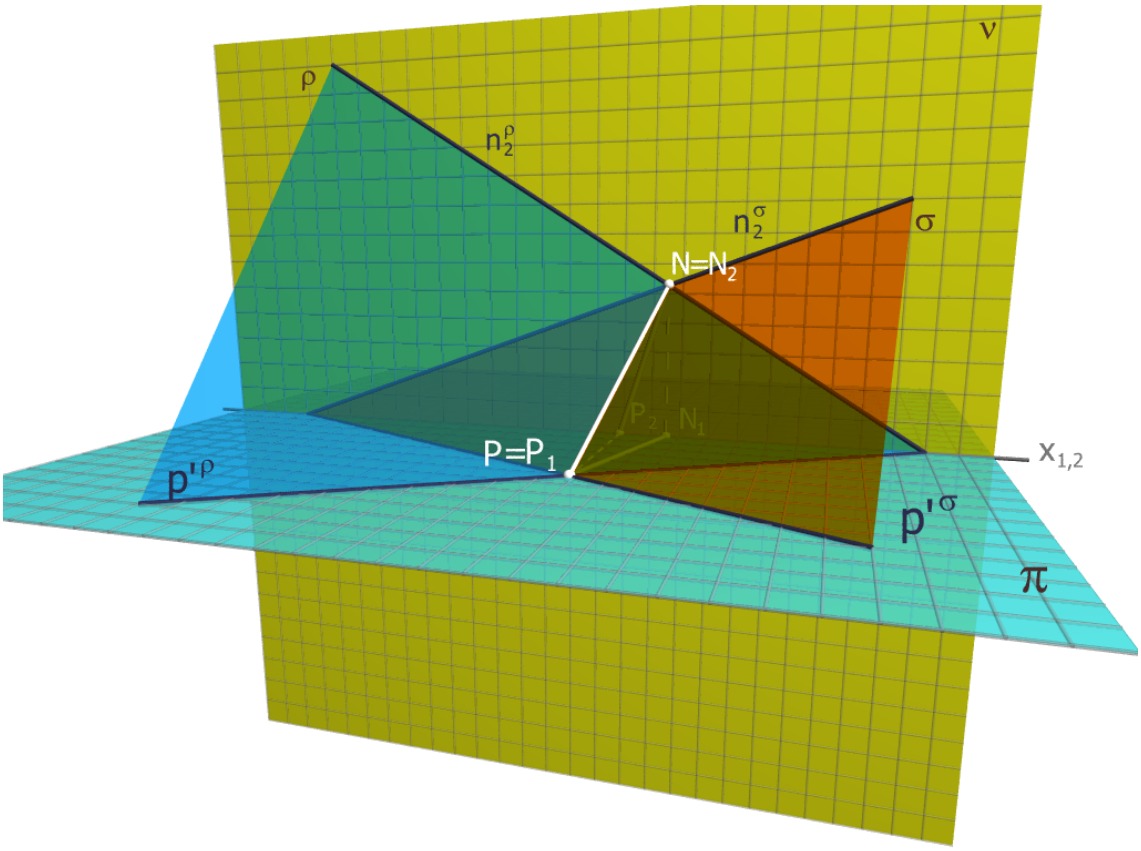
Pokud se kolmice neprotínají, jsou dané přímky mimoběžné.



Obrázek 9: Zobrazení mimoběžných přímek

3.2.2 Dvojice rovin

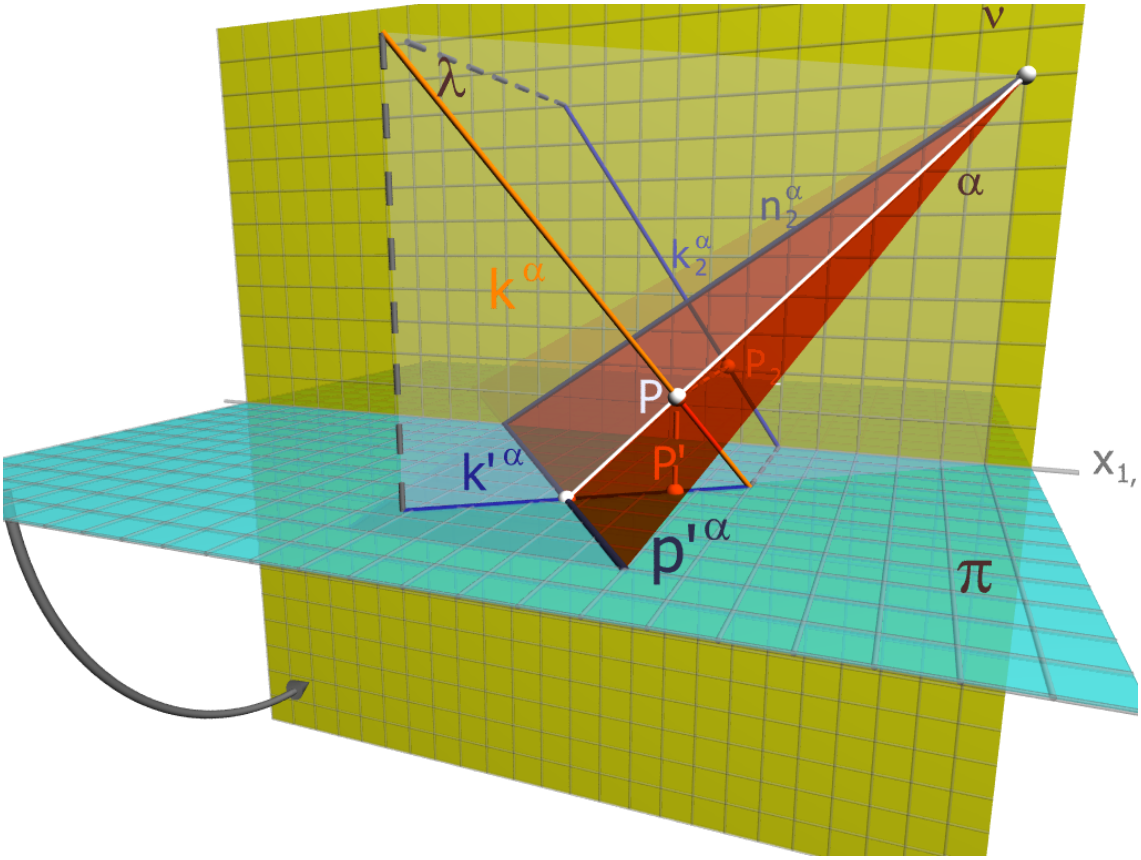
Dvě roviny mohou být buď rovnoběžné, různoběžné a nebo incidentní. Já jsem v této práci vykresloval jen případ dvou různoběžných rovin. Jejich průnikem je přímka. Každou rovinu jsem určil třemi různými body, tak jako tomu bylo v kapitole 3.1.3. Vykreslil jsem stopy těchto rovin a také jejich průsečíky. Přímka spojující tyto průsečíky je hledaná přímka průniku. Pro úplnost scény jsem ještě vykreslil průměty této přímky.



Obrázek 10: Zobrazení průniku rovin

3.2.3 Průsečík přímky s rovinou

Tak jako v předchozích příkladech jsem rovinu určil třemi jejími body a sestrojil jsem její stopy. Přímku jsem pro jednoduchost volil kolmou k rovině a k jejímu vykreslení jsem využil makro pro tvorbu kolmé přímky popsané v kapitole 2.9. Dále jsem vykreslil průměty této přímky a pomocné čáry k této přímce.



Obrázek 11: Zobrazení průsečíku přímky a roviny

Pro konstrukci průsečíku přímky s rovinou je potřeba zvolit vhodnou pomocnou rovinu λ, která přímku obsahuje.

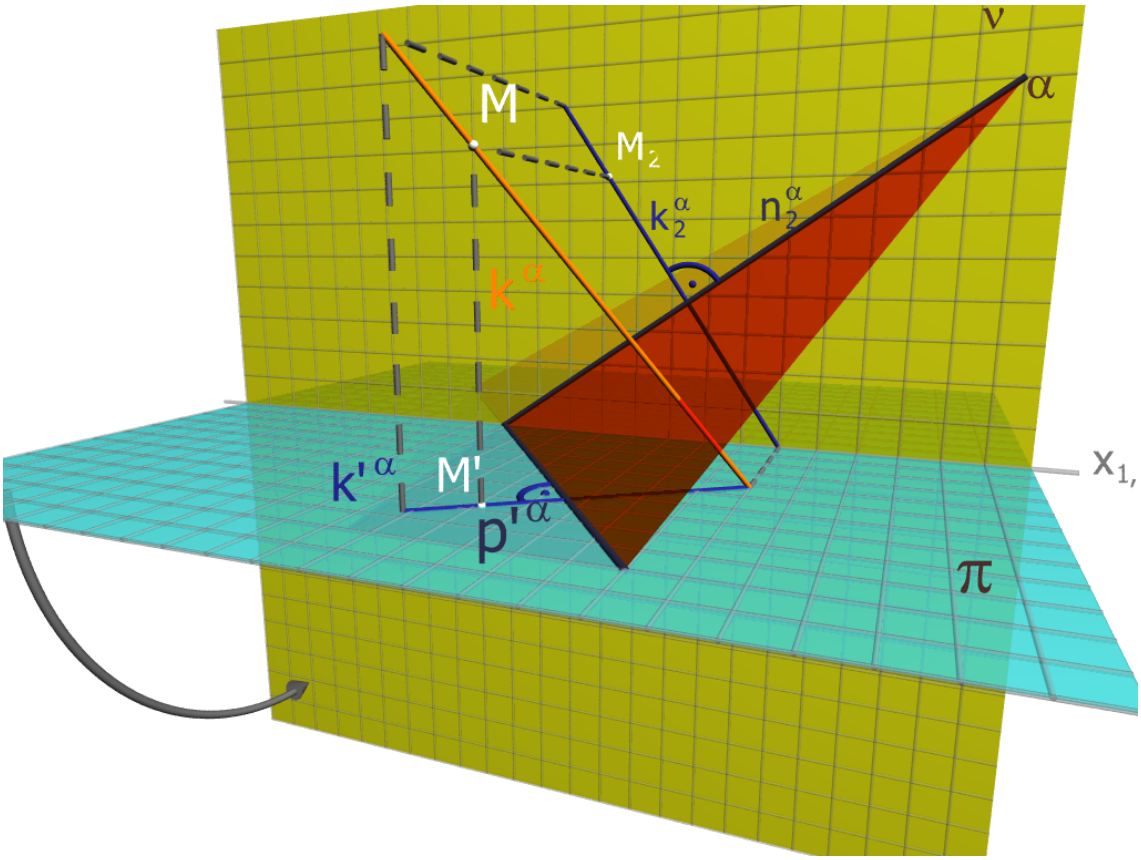
```
//------------------------- rovina lambda ------------------------
polygon {5, <Ph1, Ph2, Ph3>
            <Ph1, 0, Ph3>
            <Px1, 0, Pz1>
            <Px1, Py1, Pz1>
            <Ph1, Ph2, Ph3> pigment {Gray60 filter 0.7} }
```

Poté, co jsem zvolil vhodnou pomocnou rovinu, jsem tento příklad řešil podobně jako příklad průniku dvou rovin 3.2.2. Výsledná průnik rovin α a λ je vykreslen bílou barvou. Na závěr jsem úlohu řešil jako v případě průsečíku dvou přímek, viz. 3.2.1.

3.3 Metrické úlohy

3.3.1 Přímka kolmá k rovině jdoucí daným bodem

Z konstrukčního hlediska se jedná o jednu z lehčích úloh Mongeova promítání. Pro přímku kolmou k rovině platí, že její půdorys je kolmý k půdorysné stopě roviny a její nárys je kolmý k nárysné stopě roviny. Tyto kolmice ke stopám dané roviny se vedou z jednotlivých průmětů daného bodu. Celá scéna vypadá potom následovně.



Obrázek 12: Zobrazení přímky kolmé k rovině jdoucí daným bodem

Při popisu scénu jsem vycházel z textové souboru popisujícího průsečík přímky s rovinou popsaného v kapitole 3.2.3. Odmazal jsem ale pomocnou rovinu λ samotnou přímku tvořící průsečík rovin λ a α a všechny body a popisky s těmito objekty spojené. Nově jsem přidal bod M, jehož souřadnice jsem dopočítal následovně.

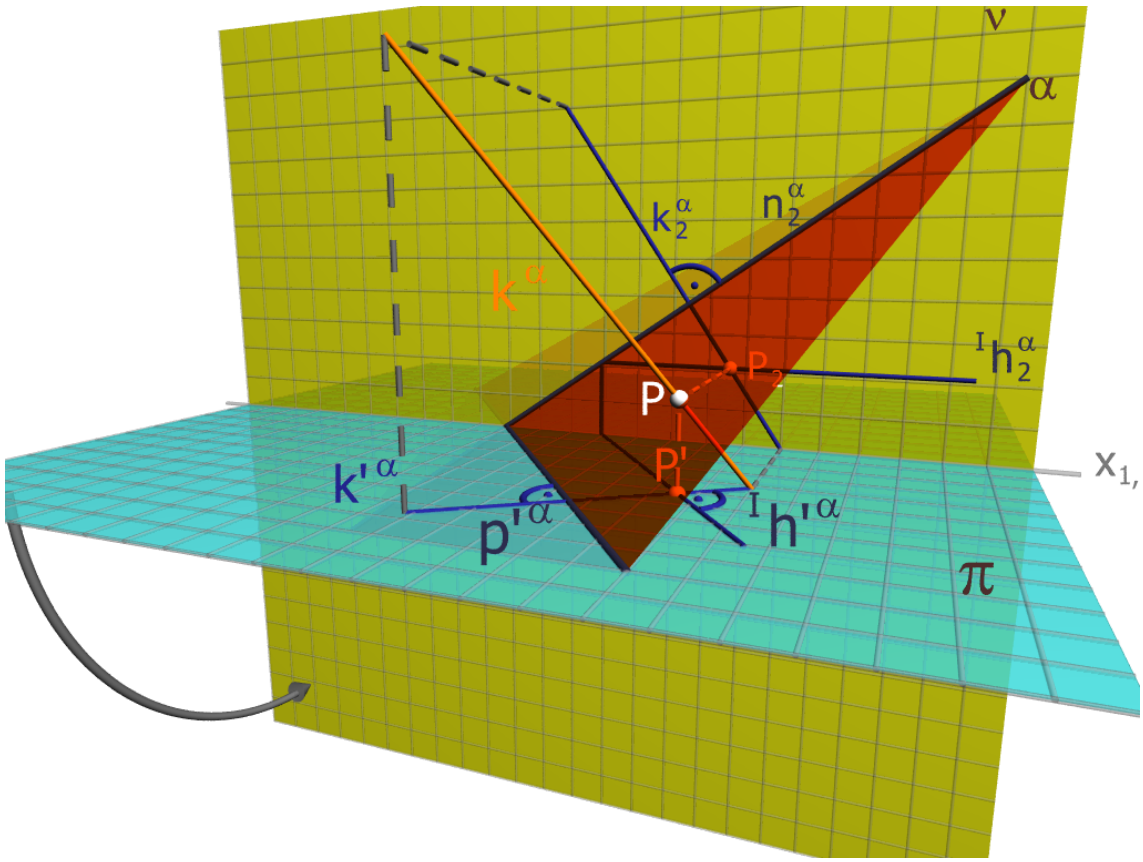
```
//-------------------------- souradnice bodu M --------------------------
#declare Sm = 0.075;
#declare Pm1 = Pp1 + Sm*vs1;
#declare Pm2 = Pp2 + Sm*vs2;
#declare Pm3 = Pp3 + Sm*vs3;
//-------------------------- bod M --------------------------
bod(Pm1, Pm2, Pm3, 0.05, White)
Pomocna_cara_vodorovna(Pm1, Pm2, 0, Pm1, Pm2, Pm3, 0.01, Pomoc)
Pomocna_cara_svisla(Pm1, 0, Pm3, Pm1, Pm2, Pm3, 0.01, Pomoc)
bod(Pm1, Pm2, 0, 0.05, White)
bod(Pm1, 0, Pm3, 0.05, White)
```

Parametr `Sm` určuje, jak bude bod M posunutý na přímce, jejíž parametrická rovnice je nadefinována hned pod tímto parametrem. Po zjištění těchto souřadnic jsem již obvyklým způsobem vykreslil bod M, jeho průměty a pomocné čáry spojující tento bod se svými průměty. Dále jsem zde použil značku pro pravý úhel popsanou v první kapitole. Tyto značky jsem ale musel příslušným způsobem otočit a posunout, pomocí příkazů `rotate` a `translate`.

```
//-------------------------- pravý úhel --------------------------
object {pravy_uhel(0.5, 0.04, Prumet)
        rotate<90, 0, 31> translate<1.7, 1.9, 0>}
object {pravy_uhel(0.5, 0.04, Prumet)
        rotate<0, 135, 0> translate<1.25, 0, -3.2>}
```

3.3.2 Rovina kolmá k přímce jdoucí daným bodem

Opět jsem vycházel z textového souboru popisujícího scénu průsečíku přímky a roviny (kapitola 3.2.3). Jako bod, kterým se má rovina, kolmá k přímce, zkonstruovat, jsem zvolil bod P zvýrazněný na obrázku bílou barvou. Pro znázornění konstrukce jsem do scény přidal průměty hlavní přímky první osnovy jdoucí bodem P.

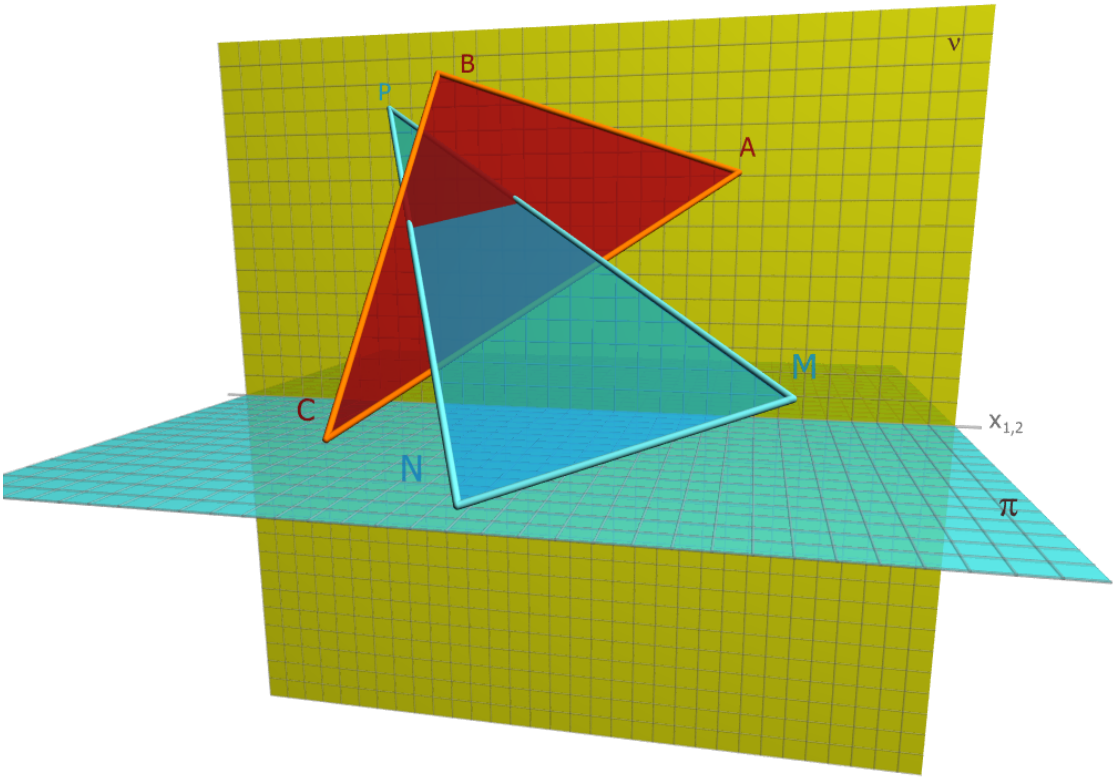


Obrázek 13: Zobrazení roviny kolmé k přímce jdoucí daným bodem

3.4 Aplikace základních úloh

3.4.1 Průnik trojúhelníků

Pro vykreslení scény jsem vycházel ze zadaných souřadnic dvou trojúhelníků ABC a PMN. Poté jsem pomocí příkazu `polygon` vykreslil samotné trojúhelníky. Pro větší přehlednost jsem se rozhodl vykreslit ještě hrany spojující jednotlivé vrcholy trojúhelníků a také přímo zadané body tvořící tyto vrcholy. Na závěr jsem už jen zvolil barevné schéma a scénu popsal příslušnými popisky.



Obrázek 14: Zobrazení průniku trojúhelníků

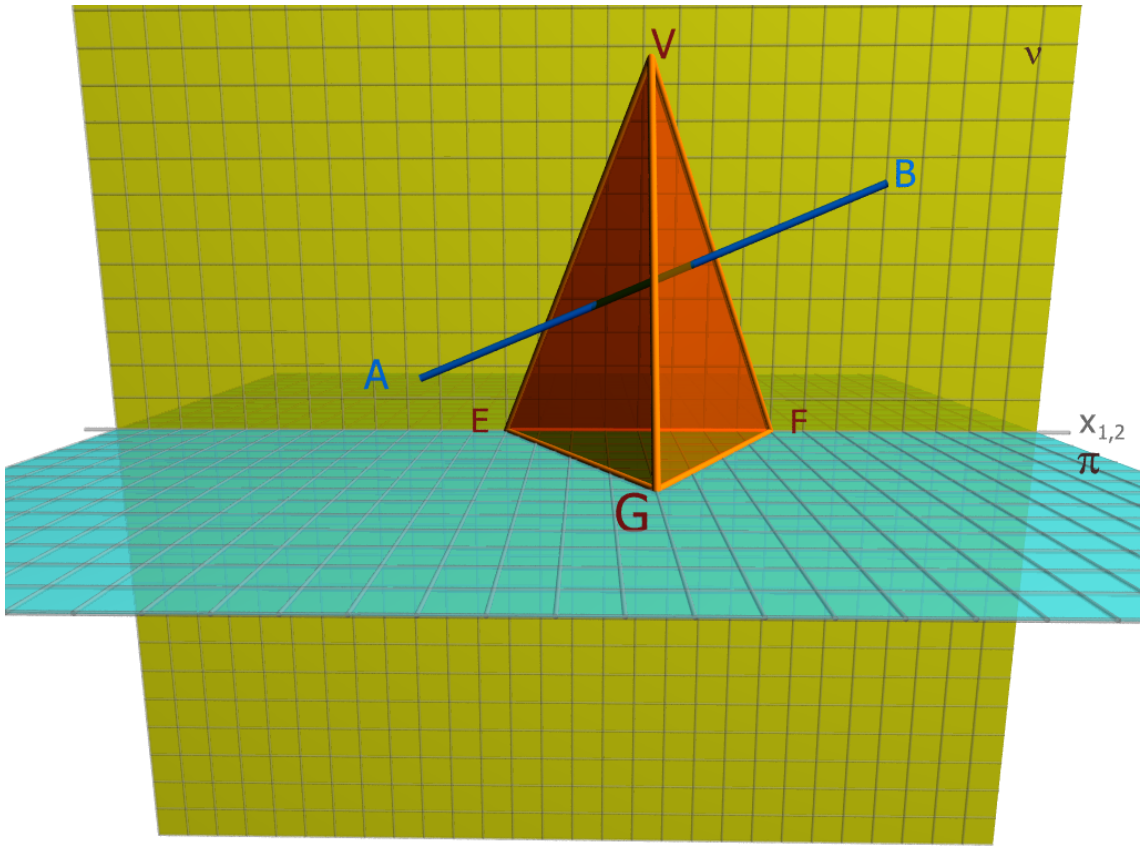
3.4.2 Průsečík přímky s jehlanem

U této scény jsem na začátku určil souřadnice bodů podstavy jehlanu a jeho vrcholu. A také souřadnice dvou bodů určujících přímku. Samotný jehlan jsem

vykreslil jako těleso složené z ploch, které jsou určeny vždy příslušnými body. Například boční stěna EFV:

```
polygon{4 <Ex1, Ey1, Ez1>
          <Fx1, Fy1, Fz1>
          <Vx1, Vy1, Vz1>
          <Ex1, Ey1, Ez1>
          pigment {color teles2} }
```

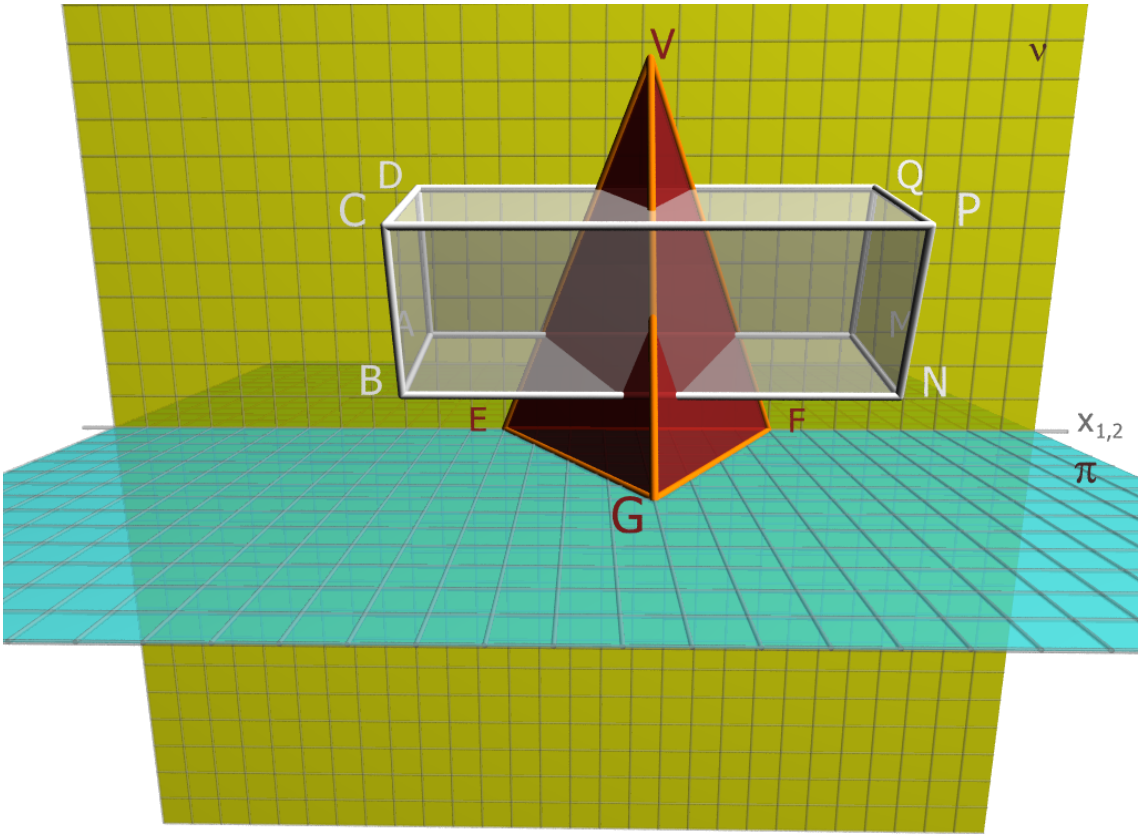
Pro větší přehlednost jsem ještě vykreslil příslušné hrany jehlanu a body ve vrcholech jehlanu. Samotnou přímku jsem vykreslil jako úsečku spojující dva zadané body. Na závěr jsem opět zvolil barevné schéma a scénu popsal.



Obrázek 15: Zobrazení průsečíku přímky s jehlanem

3.4.3 Průnik hranolu a jehlanu

Tak jako v předchozích scénách jsem nejprve zadal souřadnice vrcholů těles, poté vykreslil hrany, vrcholy a stěny těchto těles. Celá scéna potom vypadá následovně.



Obrázek 16: Zobrazení průniku hranolu a jehlanu

3.4.4 Řez rotačního kuželu rovinou

Jako poslední scénu jsem se rozhodl zobrazit řez rotačního kuželu rovinou. Kužel jsem vykreslil pomocí následujících příkazů, kde souřadnice bodu A představují vrchol jehlanu a souřadnice bodu B střed podstavy.

```
cone { <Ax1, Ay1, Az1>, 1.5,
       <Bx1, By1, Bz1>, 0
       pigment {color Kuzel} }
```

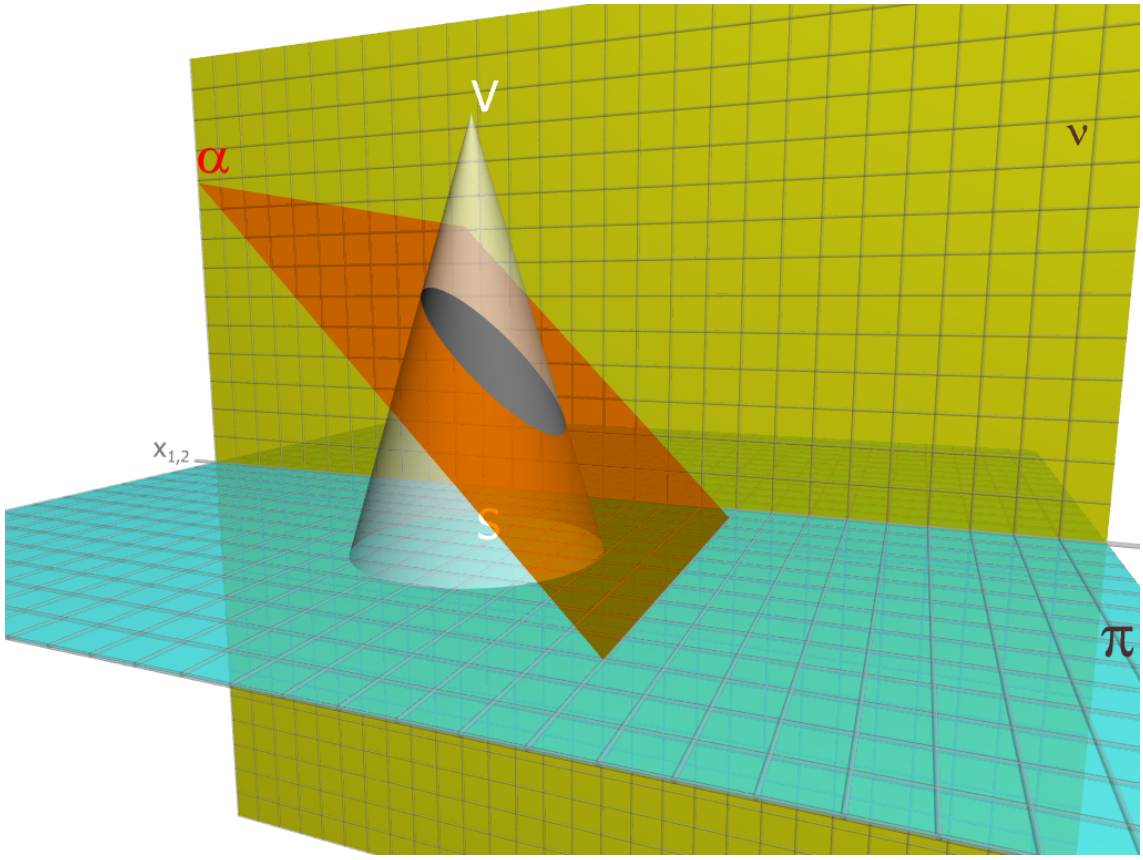
Rovinu jsem vytvořil pomocí makra pro kreslení rovin.

```
rovina (Px1, Py1, Pz1, Mx1, My1, Mz1, Nx1, Ny1, Nz1, -5.5, Rez)
```

Body P, M, N jsou tři body, které jsem si určil pro vytvoření této roviny. Samotný řez jsem se rozhodl ještě zvýraznit černou barvou.

```
difference {
   object{
      union{
         cone {<Ax1, Ay1, Az1>, 1.5,
               <Bx1, By1, Bz1>, 0 pigment {color Kuzel} }
         rovina (Px1, Py1, Pz1, Mx1, My1, Mz1, Nx1, Ny1, Nz1, -5.5, Rez)
      }
   }
   object{
      intersection {
         cone {<Ax1, Ay1, Az1>, 1.5,
               <Bx1, By1, Bz1>, 0 }
         rovina (Px1, Py1, Pz1, Mx1, My1, Mz1,
                 Nx1, Ny1, Nz1, -5.5, Black)
      }
   }
}
```

Nejprve jsem vytvořil dva objekty. První je tvořen spojením kuželu a roviny a druhý je tvořen průnikem těchto těles. Když jsem z těchto objektů potom vytvořil rozdíl, tak scéna vypadala následovně.



Obrázek 17: Zobrazení řezu rotačního kuželu rovinou

Závěr

V této práci jsem řešil vybrané problémové úlohy z Mongeova promítání, snažil jsem se o to aby zobrazovaná scéna byla co nejnázornější a aby si při pohledu na ni mohl student udělat lepší představu, jak obraz ve skutečnosti vypadá. Zdrojové kódy jednotlivých scén jsem do práce vypisoval ve zkrácené tvorbě a jen slovně jsem okomentoval, jak jsem scénu tvořil. Originální textové soubory, popisující scénu, jsou k dispozici na přiloženém CD.

Celá práce by mohla sloužit jako návod k tomu, jak vytvořit pomocné obrázky, které by se při výuce Mongeova promítání mohli studentům promítat. Obrázky jsem se snažil tvořit takovým způsobem, aby si je každý mohl rychle a jednoduše upravit dle svých potřeb (například změnou souřadnic umístění kamery, objektů, průhlednost objektů atd.).

V seznamu přiložené literatury jsou k dispozici především internetové stránky, ze kterých jsem čerpal, zabývající se tvorbou scén v progamu POV-Ray. Všem, kteří by se chtěli o práci v tomto programu dovědet něco více, a kteří by chtěli sami zkusit scény vytvářek, tyto stránky vřele doporučuji.

Literatura

[1] Urban A., Deskriptivní geometrie 1, Státní nakladatelství technické literatury, Praha 1965

[2] Internetová stránka, ze které je program POV-Ray volně ke stažení
http://www.povray.org/

[3] Internetové stránky Andrey a Friedricha A. Lohmüller
http://f-lohmueller.de/index.htm

[4] Internetové stránky pana Jiřího Doležala zabývající se mimojiné počítačovou grafikou
http://mdg.vsb.cz/jdolezal/Pgrafika/Pgrafika.html

[5] Internetová stránka, s výpisem předdefinovaných barev
http://www.buckosoft.com/ dick/pov/colors.php

[6] Server Grafika on-line
http://www.grafika.cz/art/3d/povray.html