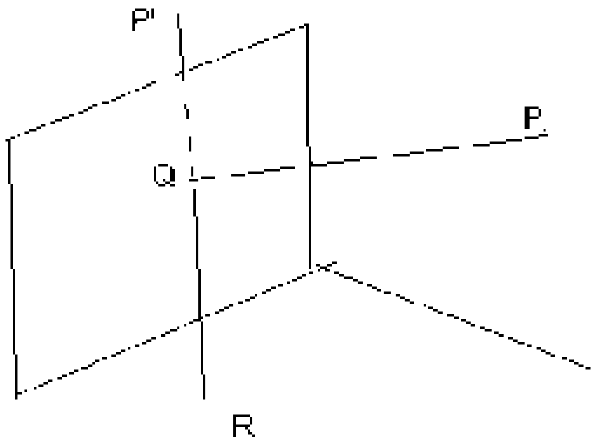
1 Mirrors: More Problems from Nature

1.1 Discussion.

What do you see when you look at something in a mirror? You see the mirror image of it! Mathematically, the object is reflected in the plane of the mirror. It is an interesting problem to draw the reflection of an object in a Maple worksheet. Just how would you do that? For example, in the diagram below, P is a point on the object and R is your eye. How do you determine the point Q that your eye sees when you look at the reflection of P in the mirror? The answer is that Q is the point on the mirror which also lies on the line from R to P', the reflection of P through the mirror. P' is defined mathematically as the point on the opposite side of the mirror such that the segment PP' is perpedicular to the mirror and is bisected by the mirror. So the point Q is a convex combination of P' and R. The equation is $Q = s\,R + (1 - s)\,P'$. If we take the plane of the mirror to be the xz-plane (y=0), then we can determine s by solving the equation in the second coordinate (indicated by a "2" subscript).

$$0 = Q_2 = s\,R_2 + (1 - s)\,P_2\,' \text{ or } s = \frac{-P_2\,'}{R_2 - P_2\,'}$$

Since $P_2\,' = -P_2$, the reflection of P_2 in the yz plane, we have that $s = \frac{P_2}{R_2 + P_2}$.

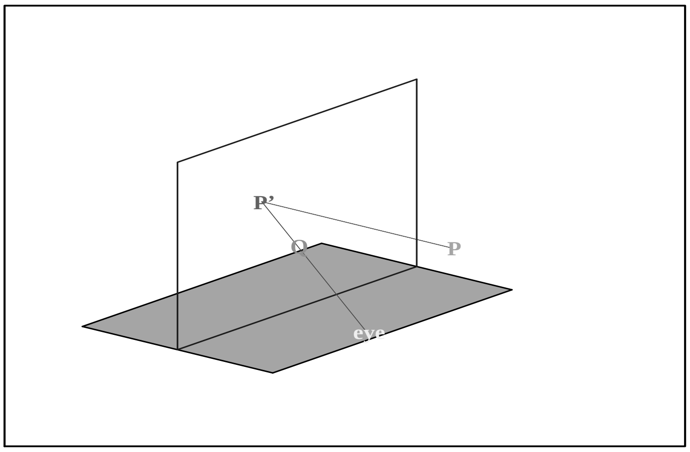


1.1.1 Diagram "light1"

```
miror:=plots[polygonplot3d]([[-2,0,0],[2,0,0]
,[2,0,3],[-2,0,3]],
style=wireframe,thickness=3,color=blue):
flor:=plots[polygonplot3d]([[-2,-1,0],[2,-1,0],[2,1,0],[-2,1,0]],
style=patch,thickness=3,color=tan):
pointP:=plots[textplot3d]([-1,1,1,` P`], font=[TIMES,BOLD,24]):
refP:=plots[textplot3d]([-1,-1,1,` P'`],
font=[TIMES,BOLD,24]):plots[display]([miror,flor,pointP,refP]):
eye:=plots[textplot3d]([2,2,1,` eye`], font=[TIMES,BOLD,24]):
P_refP:=plots[polygonplot3d]([[-1,1,1],[-1,-1,1]],
style=wireframe,color=red):
```

```
eye_refP:=plots[polygonplot3d]([[2,2,1],[-1,-1,1]],
style=wireframe,color=red):
pointQ:=plots[textplot3d]([0,0,1,` Q`], font=[TIMES,BOLD,24]):
light1:=plots[display]([miror,flor,pointP,refP,eye,P_refP,
eye_refP,pointQ],orientation=[40,60]):
```

```
light1;
```



We can define a word 'image' to return this point Q given the points P and R.

```
image := proc(p,r)
local s,q,refp;
s := p[2]/(r[2]+p[2]);
refp :=[p[1],-p[2],p[3]];
q := s*r+(1-s)*refp;
end:
```

Test this word out by taking p to be at [0,2,1] and r at [1,1,1]. In this case, the reflection of p through the xz plane is refp = [0,-2,1], the scalar s is 2/(1+2), and so the image of p on the mirror y=0 when the observer r is at [1,1,1] is 2/3*[1,1,1]+ 1/3*[0,-2,1] = [2/3, 0, 1].

```
p := [0,2,1]; r:=[1,1,1];
q := image(p,r);
```

$$p := [0,\ 2,\ 1]$$

$$r := [1,\ 1,\ 1]$$

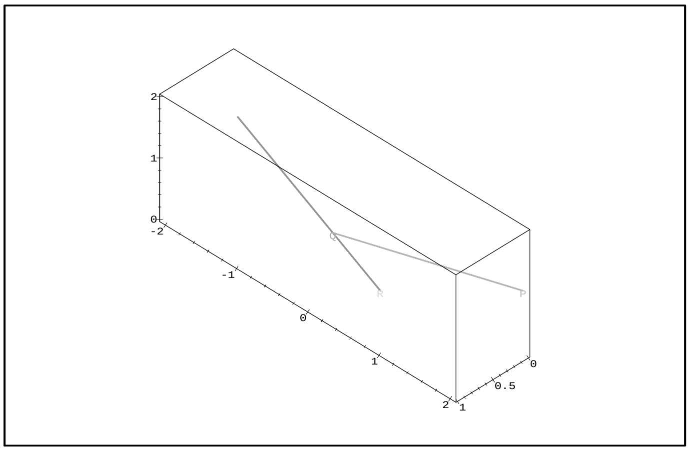
$$q := [\frac{2}{3},\ 0,\ 1]$$

That's right. Now we can draw the picture.

```
pl1 :=
plots[polygonplot3d]({[p,q],[r,[p[1],-p[2],p[3]]]},
thickness=3,scaling=constrained,axes=boxed):
```

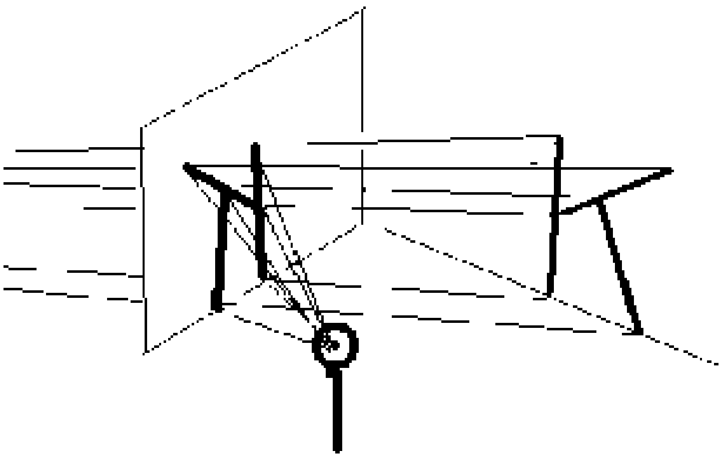
```
pl2 :=plots[textplot3d]({[op(q),`Q`],
[op(r),`R`],[op(p),`P`]},
align=BELOW):
```

```
plots[display]([pl1,pl2]);
```



1.1.1 *Problem: Draw the letter K and its reflection in a mirror.*

For the sake of simplicity, let us assume that the letter K is standing in the yz plane on the y-axis, with one base at say [0,1,0] and the other at [0,2,0]. Our mirror will be a square in the xz plane with base [[0,0,0],[2,0,0]]. We, the observer will be standing at a point (a,b,c) to be determined later. The picture looks like this:



First define the mirror and the letter k.

```
mir :=
[[0,0,0],[2,0,0],[2,0,2],[0,0,2]];
```

$$mir := [[0, 0, 0], [2, 0, 0], [2, 0, 2], [0, 0, 2]]$$

```
mirror :=
plots[polygonplot3d](mir,style=patch,color=gray):
```

```
kay :=
[[0,1,0],[0,1,1]],[[0,1,.5],[0,1.5,1]],
[[0,1,.5],[0,1.5,0]];
```

$$kay := [[0, 1, 0], [0, 1, 1]], [[0, 1, .5], [0, 1.5, 1]], [[0, 1, .5], [0, 1.5, 0]]$$

```
ka
:=plots[polygonplot3d]({kay},thickness=3,color=red):
```

Now the image of the letter k in the mirror is a function of the position (a,b,c) of the observer.

```
refkay := proc(a,b,c)
local i;
seq(map(image,kay[i],[a,b,c]),i=1..nops([kay]));
end:
```

So let's pick an observation point ob

```
ob := [2,3,1];
```

$$ob := [2, 3, 1]$$

obs is a blue stick located at the observation point.

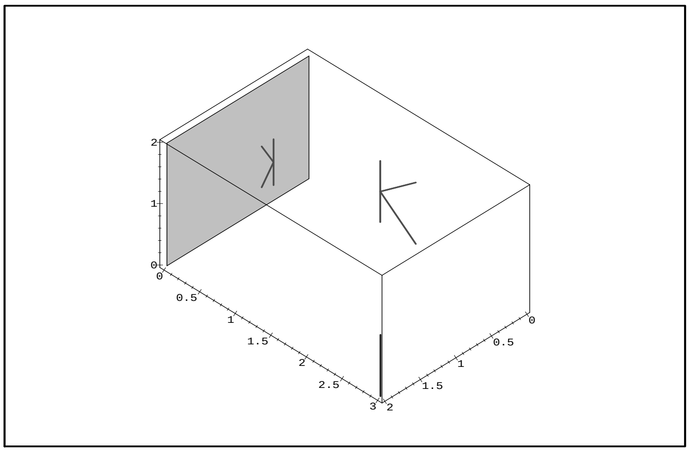
```
obs :=
plots[polygonplot3d]([ob,[ob[1],ob[2],0]],color=blue,thickness=3):
```

reka is the image of the letter k in the mirror from the observation point ob.

```
reka
:=plots[polygonplot3d]({refkay(op(ob))},thickness=3,
color=red):
```

We can see what this looks like with display.

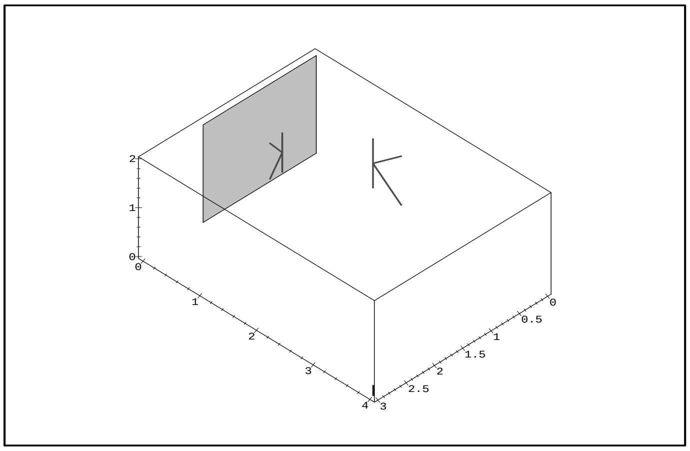
```
plots[display]([obs,mirror,ka,reka],scaling=c
onstrained,axes=boxed);
```



Looks fine. Copy down the input cells from above to make a word to draw the mirror, the letter, and its image in the mirror as seen from the observer.

```
mirimage := proc(ob)
local obs, reka;
obs :=
plots[polygonplot3d]([ob,[ob[1],ob[2],0]],color=blue,thickness=3):
reka :=plots[polygonplot3d]({refkay(op(ob))},thickness=3,color=red):
plots[display]([obs,mirror,ka,reka],scaling=constrained,axes=boxed);
end:
```

```
mirimage([3, 4,.2]);
```



1.1.2 *Exercise: Make a movie showing what happens to the reflection as we move the observer around in a circle .*

```
movie :=
[seq(mirimage([2+cos(j*Pi/5),2+sin(Pi*j/5),1]),j=1..10)]:
```

```
display(movie,insequence=true);
```

1.2 Questions:

1.2.1 *Question: Suppose we had another mirror in a vertical plane perpendicular with the first mirror with its base on the segment [[3,1,0],[3,3,0]]. If an observer at [2,2,1] looked in the second mirror, what would he see (besides himself)? Model this.*

1.2.2 *Question If you have two perfect mirrors, and hold them just right, you can see down an infinite corridor. Even with two good mirrors, you can see a large number of reflections of reflections. Observe this and describe what you see exactly. As an experiment, hold up a transparency with a letter k written on it. How hard would this be to model?*

1.2.3 *Question: If you have three mirrors which form a corner of a box and shine a beam of light almost into the corner, what happens to the beam?*

1.2.4 *Problem: Model a kaleidoscope.*

1.2.5 *Problem: Here's a two dimensional problem. Suppose you shine a light down a mirrored angle. How many times will it bounce before it comes out? (Or will it come out?).*

1.3 Some solutions

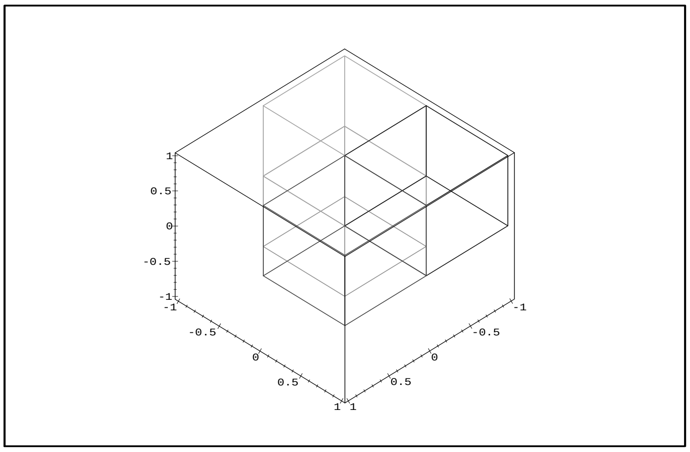
A solution to question 3.

If we shine the light from say [1,1,1] almost into the corner [0,0,0], then it will be reflected in some succession off of each coordinate plane, effecting a sign change in the corresponding component of its velocity vector with each bounce.. Upon the third bounce, its velocity will be the negative of its original velocity and so it will be moving parallel to its original direction but in the opposite direction. We can model this. First we need our cube word.

```
cube := proc (shift,clr)
local v1, v2, v3, v4, v5, v6, v7, v8, front, back, left, right,
bottom, top, box;
v1 := [0, 0, 0]+shift; v2 := [1, 0, 0]+shift; v3 := [1, 1, 0]+shift;
v4 := [0, 1, 0]+shift; v5 := [0, 0, 1]+shift; v6 := [1, 0, 1]+shift;
v7 := [1, 1, 1]+shift; v8 := [0, 1, 1]+shift;
front := [v1, v2, v3, v4]; back := [v5, v6, v7, v8];
left := [v4, v8, v5, v1]; right := [v3, v2, v6, v7];
bottom := [v3, v4, v8, v7]; top := [v1, v2, v6, v5];
box := [front, back, left, right, top, bottom];
plots[polygonplot3d](box,color=clr,scaling=constrained,
style = wireframe,axes=boxed)
end:
```

We propose to construct the path of the light ray in the cube by first constructing the 'straightened out' path we would get if the ray continued on through the walls of the cube. This straight path, which could be called the 'virtual path' of the light ray, can then be fairly easily 'wound back up' by reflecting the points where the straight path crosses the coordinate planes into the original cube.

```
plots[display]([cube([0,0,0],red),cube([-1,0,
0],blue),
cube([-1,-1,0],tan),cube([-1,-1,-1],green)]);
```



So assume our light ray starts at $p = [.9, 1, .9]$ (in the red cube) and heads toward $q = [-1, -.5, -.25]$ (in the green cube). Along the way it will successively cross the xy, yz, and zx planes in some order. We can easily determine the crossing points and order of crossing by parameterizing the segment from p to q and solving three equations.

```
p := [9/10,1,9/10]:  q := [-1,-1/2,-1/4]:
```

Using t as the parameter, a general point r on the segment pq looks like

```
r := expand(t*q+(1-t)*p);
```

$$r := [\frac{9}{10} - \frac{19}{10}\,t,\ 1 - \frac{3}{2}\,t,\ \frac{9}{10} - \frac{23}{20}\,t]$$

where $t \in [0, 1]$. Now to get the crossing points of the yz, xz, and xy planes solve the equations.

```
tyz := solve(r[1]=0,t);
txz := solve(r[2]=0,t);
txy := solve(r[3]=0,t);
```

$$tyz := \frac{9}{19}$$

$$txz := \frac{2}{3}$$

$$txy := \frac{18}{23}$$

By inspection, we have tyz < txy < txz, so the yx, xy, and xz planes are crossed in that order. The crossing points are obtained by substitution.

```
c1 := subs(t=tyz,r);
c2 := subs(t=txz,r);
c3 := subs(t=txy,r);
```

$$c1 := [0, \frac{11}{38}, \frac{27}{76}]$$

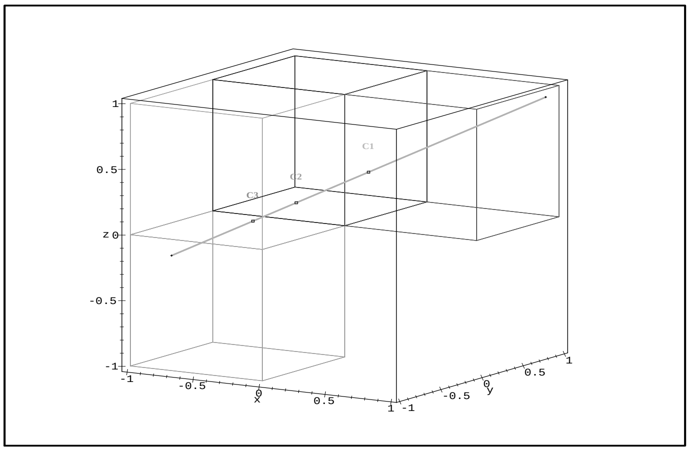
$$c2 := [\frac{-11}{30}, 0, \frac{2}{15}]$$

$$c3 := [\frac{-27}{46}, \frac{-4}{23}, 0]$$

In terms of the colored cubes, the virtual light ray traverses the red cube, then crosses the yz plane at c1 into the blue cube, traverses the blue cube, then crosses the xz plane at c2 into the tan cube, traverses that, and finally crosses the xy plane at c3 and goes into the green cube.

Here is a plot showing the segment pq and the three crossing points c1,c2, c3.

```
ptplt :=
plots[pointplot3d]({p,q},color=black,thickness=3):
seg := plots[polygonplot3d]([p,q],thickness=3):
pts:=plots[textplot3d]({[op(c1 +[0,0,.2]),`C1`],[op(c2
+[0,0,.2]),`C2`],[op(c3 +[0,0,.2]),`C3`]},font=[TIMES,BOLD,10]):
ptplt2 := plots[pointplot3d]([c1,c2,c3],color=blue,symbol=box):
plots[display]([pts,
ptplt2,ptplt,seg,cube([0,0,0],red),cube([-1,0,0],blue),
cube([-1,-1,0],tan),cube([-1,-1,-1],green)],labels=[x,y,z],orientation
=[-58,78]);
```



The actual light ray traverses the red cube, bounces off of the yz plane at c1, traverses the red cube again, then bounces off the xz plane at the reflection rc2 of c2 through the yz plane, traverses the red cube again, then bounces off the xy plane at the second reflection rrc3 of c3 first through the yz plane and then through the xz plane.

Finally, it strikes the wall x=1 of the red (original) cube at -q = [1,.5,.25] at t=1.

```
rc2 :=[-c2[1],c2[2],c2[3]];
rrc3 := [- c3[1],-c3[2],c3[3]];
mq := -q;
```

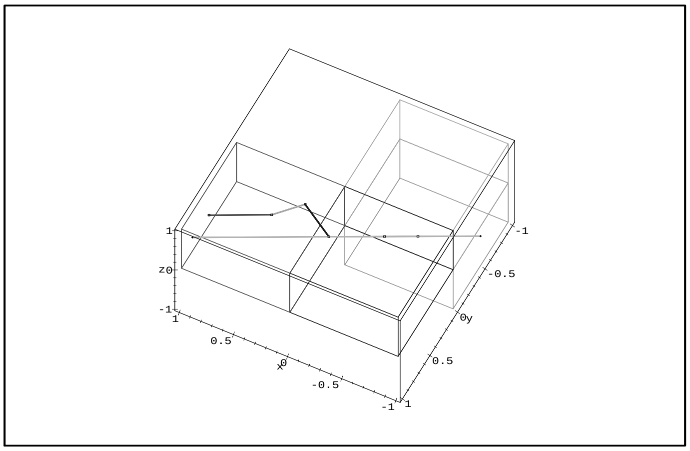
$$rc2 := [\frac{11}{30}, 0, \frac{2}{15}]$$

$$rrc3 := [\frac{27}{46}, \frac{4}{23}, 0]$$

$$mq := [1, \frac{1}{2}, \frac{1}{4}]$$

Here is a plot showing the actual path of the light ray in the cube and the virtual path..

```
seg2 :=
plots[polygonplot3d]([c1,rc2],thickness=3,color=blue):
seg3 := plots[polygonplot3d]([ rc2,rrc3], thickness=3, color=tan):
seg4 := plots[polygonplot3d]([rrc3,mq], thickness=3,color=red):
ptplt3 := plots[pointplot3d]([rc2,rrc3,mq],color=blue,symbol=box):
plots[display]([ptplt3,ptplt2,ptplt,seg,seg2,seg3,seg4,
cube([0,0,0],red),cube([-1,0,0],blue),cube([-1,-1,0],tan),
cube([-1,-1,-1],green)],labels=[x,y,z],orientation=[117,22]);
```



From within the red box the the ray bounces off the "far wall" to the left following the thick blue path, then to the "floor" along the tan path, then back along a path parallel to its original path of entry.

1.3.1 *Problem: Define a word cornerbounce(p,q) where p is a point on the red cube with exactly one coordinate 1 and no coordinate 0, and q is a point on the green cube with exactly one coordinate -1 and no coordinate 0, which returns a plot of both the virtual and actual paths of a beam of light from p to q.*

1.3.2 *Problem: Model the path of a light beam bouncing around in an enclosed cubical room.*

Analysis: We can continue with the 'virtual ray' idea that we were using before, but with a slightly different twist. Think of the virtual light beam as parameterized by the point $p + tv$ where p is the starting point, t is the (time) parameter, and v is the velocity vector. The room is the set of all points [x,y,z] such that each coordinate is between 0 and 1. A point q on the on the actual light beam is obtained from a point $p + t\,v$ on the virtual light beam as follows: for each coordinate i, if the integer part of $p_i + t\,v_i$ is even, then q_i is the fractional part of $p_i + t\,v_i$ and if the integer part is odd, then q_i is 1 - fractional part.

The word pathpt computes q from p, t, and v.

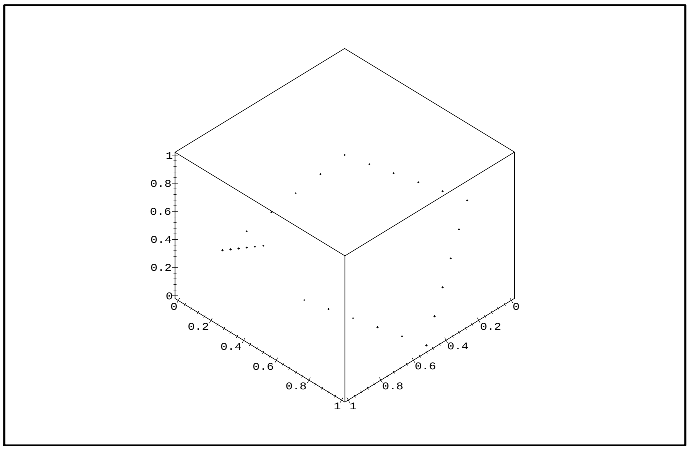
```
pathpt := proc(p,v,t)
local q,i,coord;
for i from 1 to 3 do
coord := p[i] + t*v[i];
if floor(coord) mod 2 = 0 then q[i] := frac(coord)
else q[i] := 1-frac(coord) fi od;
[q[1],q[2],q[3]]; end:


path := proc(p,v,s,n) local i;
[seq(pathpt( p,v,s*i),i=0..n)] end:
```

The word path constructs a list of points on the actual path, and the word drawpath uses spacecurve from the plots package to draw a picture of the path of the light beam in the cube.

```
drawpath := pth ->
plots[spacecurve](pth, color=black, style=point, scaling=constrained,
axes=boxed):


drawpath(
path([1/2,0,0],[.1,.05,.1],1,25));
```



```
movie := pth -> plots[display](
[seq(drawpath(pth[1..j]),
j=1..nops(pth))],insequence=true):
```

Here is a movie of the beam bouncing.

```
movie(path([1/2,0,0],[.1,.05,.1],1,50));
```

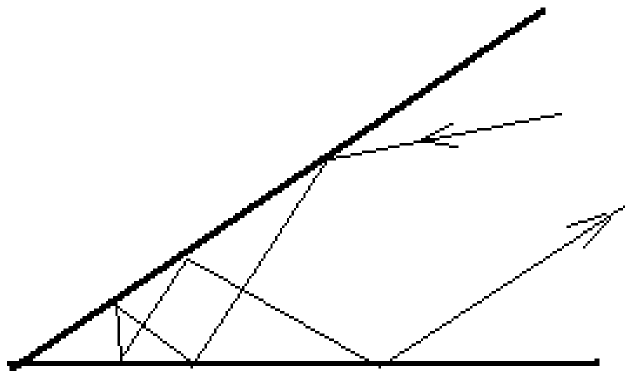
1.3.3 *Exercise: Draw a few paths in the cube. Look at them from the top of the cube by grabbing it, pulling it so that you only see the top of the cube, and choosing Redraw from the menu. Describe what you see. How does it relate to the problem of modeling the path of a light beam in a square?*

1.3.4 *Problem. A path in the cube is called a closed reflective path if after a finite number of bounces, the light beam begins to repeat its path in the cube. Find some closed reflective paths in the cube. Are there any which bounce off of each face of the cube at exactly one place?*

A discussion and solution(s) to question 5.

Recall the question: Suppose you shine a light down a mirrored angle. How many times will it bounce before it comes out? Or will it come out?

First we create a word "geombouncer" is written with the geometry package which will allow us to experiment with the question. "geombouncer(A,B,dir,maxbounces)" assumes that one is looking at the angle formed by two mirrors. Looking along their edges one sees the line segment OA where O is the origin and A is a specified point in the first quadrant, and a segment along the x-axis. The point P is the starting point of a ray that is directed in to the angle with direction "dir". "geombouncer" then follows the path of the ray as it reflects from the two mirrors as long as it remains to the left of the point A up to "maxbounces" reflections.



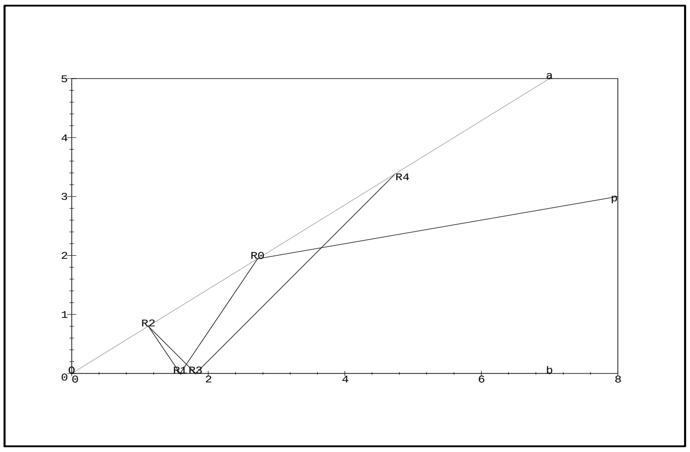
1.4 geombouncer instructions

```
geombouncer:=proc(A,P, dir,maxbounces)
local objlist,j:
geometry[point](O,[0,0]);
geometry[point](a,A):
geometry[point](c,P+dir):
geometry[point](b,[A[1],0]):
geometry[point](p,P):
geometry[line](topline,[O,a]):
geometry[segment](top,[O,a]):
geometry[line](baseline,[O,b]):
geometry[segment](base,[O,b]):
geometry[line](lnofpth0,[p,c]):
geometry[intersection](R0,lnofpth0,topline):
geometry[reflection](lnofpth1,lnofpth0,topline):
geometry[intersection](R1,lnofpth1,baseline):
geometry[dsegment](path0, [R0,R1]):
geometry[dsegment](initialpath, [p,R0]):
objlist:=NULL:
objlist:=R0,a,b,O,top(color=tan,thickness=2),base(color=tan,thickness=
2),initialpath(color=blue,thickness=2),path0(color=blue,thickness=2):
for j from 1 to maxbounces while (
geometry[coordinates](cat(R,j))[1]< A[1] and
geometry[coordinates](cat(R,j))[1]>0 ) do
geometry[draw]({objlist});
if (geometry[coordinates](cat(R,j))[2]<
geometry[coordinates](cat(R,j-1))[2]) then
geometry[reflection](cat(lnofpth,j+1),cat(lnofpth,j),baseline):
geometry[intersection](cat(R,j+1),cat(lnofpth,j+1),topline):
else
geometry[reflection](cat(lnofpth,j+1),cat(lnofpth,j),topline):
geometry[intersection](cat(R,j+1),cat(lnofpth,j+1),baseline):
fi:
geometry[dsegment](cat(path,j),[cat(R,j),cat(R,j+1)]):
```

```
if geometry[coordinates](cat(R,j+1))[1] >0 then
objlist:=objlist,cat(R,j),cat(path,j)(color=blue,thickness=2):  fi:
od:
geometry[draw]({objlist},scaling=constrained);
end:
```

```
restart;
```

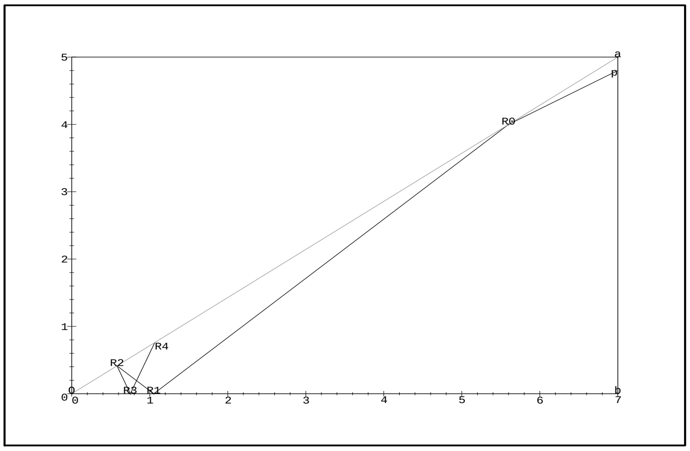
```
geombouncer([7,5],[8,3],[-1,-.2],10);
```



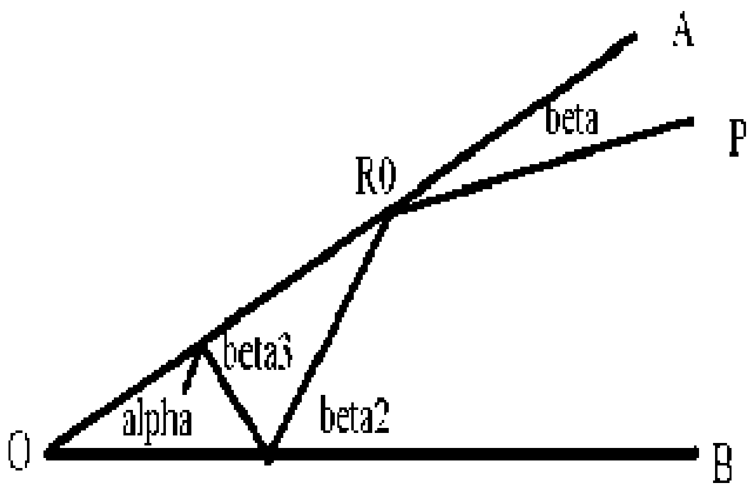
In the above example the ray emits from [8,3] in the direction [-1,-.2] and bounces five times before it leaves the mirrors.

1.4.1 *Exercise: Experiment with geombouncer to see if you can make the ray bounce a large number of times. Note that simply chosing a shallow angle won't necessarily produce a small number.*

```
geombouncer([7,5],[7,4.8],[-7,-4],10);
```



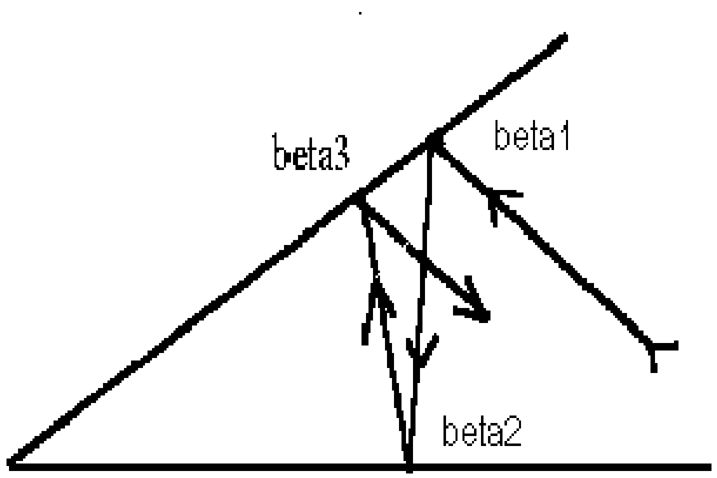
To analyze this problem we will find it convenient to use the two angles α and β as indicated in the following diagram.



That is, α is the angle between the two mirrors (which we take to be acute) and

β is the angle of incidence of the ray upon the upper mirror. Actually, what we want to look at is the sequence of incidence angles

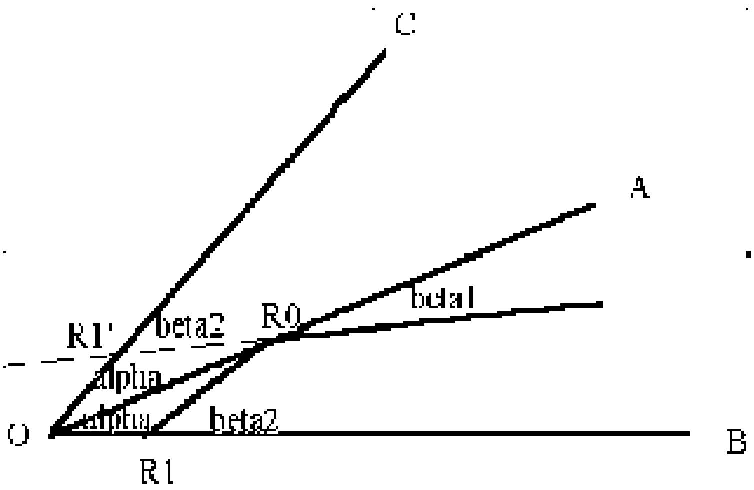
$\beta = \beta_1$, β_2, ... The basic idea is that the angles β_i are increasing in size and as soon as β_j becomes greater than or equal to $\frac{\pi}{2}$ the ray will be on its way out of the region between the mirrors



Thus we need to see how to calculate β_i for each i. This is not hard to do if we determine the reflection of the angle rather than the angle itself. Consider the following diagram in which the angle AOB is reflected through the line AO to produce the angle AOC. The initial ray strikes AO at the point R0 and angle β_1 . It is reflected, hitting the segment AO at the point R1 at angle

β_2. If the original path were continued it would meet OC at a point R1'.

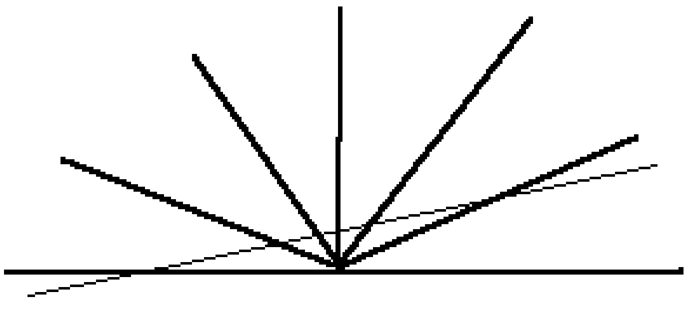
1.4.2 *Exercise: Show that, as indicated in the diagram, the angle R0R0'C is equal to* β_2 *and that* $\beta_2 = \beta_1 + \alpha$



1.4.3 *Exercise: Extend the previous argument by reflecting AOB first through AO and then through OC and use the same idea to show that* $\beta_3 = \beta_2 + \alpha$. *This* $\beta_3 = \beta_1 + 2\,\alpha$ *and in general* $\beta_n = \beta_1 + (n-1)\,\alpha$.

1.4.4 *Exercise: Argue that there are exactly as many bounces of the light beam between the two mirrors as there are points of intersection of the extended original path of the fan of rays with rays spaced exatly* α *apart opening counterclockwise from the x-axis.*

Show that the ray will start moving away from the center on that pth bounce where p is the smallest integer such that $\frac{\pi}{2} \leq \beta + (p-1)\,\alpha$



1.4.5 *Exercise: Show that there will be q bounces where q is the largest integer such that* $q\,\alpha < \pi + \beta$

1.4.6 *Exercise: Show that the light will exactly retrace its enrty route to leave if and only if* $\frac{\frac{\pi}{2}-\beta}{\alpha}$ *is a non-negative integer.*

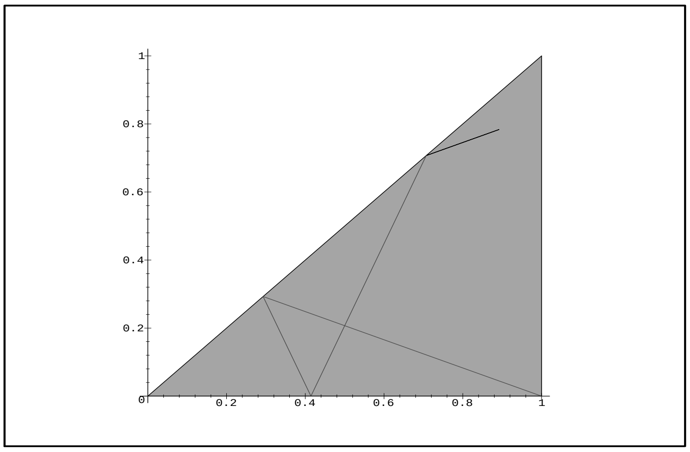
1.4.7 *Exercise: If two mirrors meet at a 30 degree angle what is the maximum possible number of reflections off of the mirrors of a light ray which is incident upon one of them?*

1.4.8 *Exercise: Adapt the instructions in the "geombouncer" word to produce an animation of the ray following its path rather than a static image of the final path.*

We used the geometry package for our diagrams earlier but there is no reason for our not writing a "bouncer" word ourselves. The following word "bouncer" takes in angles alpha and beta where theta is the angle between two lines and alpha is an angle of at most alpha which an incident ray makes with one of the legs of the angle. The procedure returns a plot of the path of the ray as it enters and bounces out of the angle between the two lines. It is followed by an an alternate discussion of the calculation of the numnber of intersections, paths that retrace themselves, etc.

```
bouncer:=proc(alpha, beta)
local
STARTPT,n,w,line,nalphaline,eqn,eqns,PTS,leg1,leg2,refpts,i,N,slns;
STARTPT:=evalf([cos(alpha),sin(alpha)]);
line:=evalf(expand(STARTPT+t*[ cos(alpha-beta), sin(alpha-beta)]));
nalphaline:=s*[cos(n*alpha) ,sin(n*alpha)];
eqn:=expand(line-nalphaline);
eqns:={eqn[1]=0,eqn[2]=0};
PTS:=NULL;
N:= floor((Pi-beta)/alpha):
if frac((Pi-beta)/(alpha))=0 then N:=N-1 fi;
for w from 1 to N+1 do
slns:=fsolve(subs(n=w,eqns),{s,t}):
PTS:=PTS,subs(slns, line):
od:  PTS:=[PTS];
leg1:=evalf([cos(alpha),sin(alpha)]);leg2:=[1,0];
refpts:=NULL:
for i from 1 to nops(PTS) do
if i mod 2 = 1 then refpts:=refpts,linalg[norm](PTS[i],2)*leg1; else
refpts:=refpts, linalg[norm](PTS[i],2)*leg2 fi; od:
plots[display](
[plots[polygonplot]([[0,0],[1,0],[1,tan(alpha)]],style=patch,color=tan
), plot([refpts]),
plots[polygonplot]([STARTPT,subs(t=.2,line),STARTPT],scaling=constrain
ed )]):
end:


bouncer(Pi/4,Pi/8);
```



The unterminated line is the incident ray and the returning path ends at its last bounce point. That is the returning ray will reflect at an angle smaller than theta, the lower left angle of the colored triangle and will therefore not hit either leg of the triangle again.

If α is the incident angle which is less than or equal to theta then the angle at which the ray rebounds at the ith bounce is $\alpha + i\,\theta$ (where the initial bounce is the "zeroth"). The condition for the ray to return exactly along its original path is that it hit one of the legs at a right angle, that is that $\alpha + n\,\theta$ is a right angle

```
eqn1:=beta+n*alpha=Pi/2;
```

```
critbeta:=solve(eqn1,beta);
```

Since $\beta \leq \alpha$ we see that n is the integer part of $\frac{\pi}{2\,\alpha}$ unless this is actually an integer in which case this value of n is one less than $\frac{\pi}{2\,\alpha}$. The latter situation only occurs when theta has the form $\frac{\pi}{2\,k}$ where k is an integer. For the moment, we ignore this possibility and have the formula

```
critbeta:=proc(alpha)
Pi/2-floor(Pi/(2*alpha))*alpha end;
```

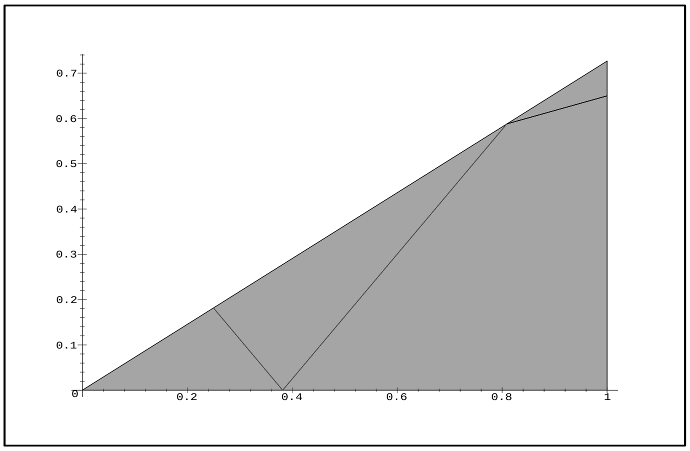
$$critbeta := \mathrm{proc}(\alpha)\, 1/2 \times \pi - \mathrm{floor}(1/2 \times \pi/\alpha) \times \alpha\, \mathrm{end}$$

Now we can draw most critical paths:

```
backbouncer:= proc(alpha) bouncer(alpha,
critbeta(alpha))end;
```

$$backbouncer := \mathrm{proc}(\alpha)\, \mathrm{bouncer}(\alpha,\, \mathrm{critbeta}(\alpha))\, \mathrm{end}$$

```
backbouncer(Pi/5);
```



Note that backbouncer does fine with angles other than those like $\frac{\pi}{6}$, $\frac{\pi}{4}$ etc.

1.4.9 *Exercise: Complete backbouncer so that it works for all positive angles less that* $\frac{\pi}{2}$.

1.4.10 *Exercise: Do an animation of backbouncer as theta changes between* $\frac{\pi}{5} - \frac{1}{4}$ *and* $\frac{\pi}{5} + \frac{1}{4}$.

1.5 Solution

```
movie:=NULL: for i from 1 to 20 do
movie:=movie,backbouncer(Pi/5-1/4+i/40):
od:plots[display]([movie],insequence=true);
```

Exercise: In the animation for the previous exercise the leg at which the ray meets perpendicularly switches several times. Where (i.e. at which angles theta) do such switches happen?

2