Fall 2021

APPLIED PHYSICS - LAB

Lab-3

- Solutions to Lab-2
- Visualization of spaces and vectors
- Position vectors
- Color Vectors

VPython documentation
VPython vector operations
Vpython basic mathematics

Solution to Lab-2:

1. Find the area of a parallelogram enclosed by the vectors `r = <2.3,3,1.4>` and `s = <3.7,3.1,2.5>`.

```
r = vector(2.3,3,1.4)
s = vector(3.7,3.1,2.5)
print( mag(cross(r,s)) )
```

2. Generate a new vector `t` on the same plane as `r` and `s`.

```
t = cross(r, cross(r,s))        # possible solution
t2 = r+s                # another possible solution
```

3. Find the projection of `t` on `r` and s.

```
proj_r = t.dot(r)
proj_s = t.dot(s)
```

4. Generate a new vector `u` which is 2 units away from `t` in x coordinate, 2.5 units in y, and 1.17 units away in z coordinates.

```
u = t + vector(2, 2.5, 1.17)
```

Implement the following codes in your own trinket.

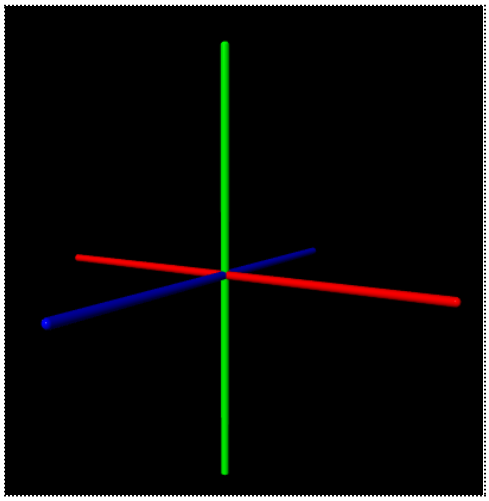
Visualization of spaces and vectors:

The last problem we solved might be a bit mind boggling for some. Let us visualize in VPython what we were actually doing. First we need to import *everything* from the visual library. We do this by writing `from visual import *` on the top of our program. Next we use the `curve()` function to generate axis lines. The appropriate method for calling a curve is the following.

```
c = curve(pos=[v1,v2], color=color.cyan, radius=0.3)
```

v1 and v2 represent the initial and final position points (as vectors) for the curve, we can add the number of these points. You can read further options for this function in the vpython documentation.

```
1  GlowScript 3.1 VPython
2  from visual import *
3
4  x_axis = curve(pos=[vector(-5,0,0),vector(5,0,0)], color=color.red, radius=0.1)
5  y_axis = curve(pos=[vector(0,-5,0),vector(0,5,0)], color=color.green, radius=0.1)
6  z_axis = curve(pos=[vector(0,0,-5),vector(0,0,5)], color=color.blue, radius=0.1)
7
```

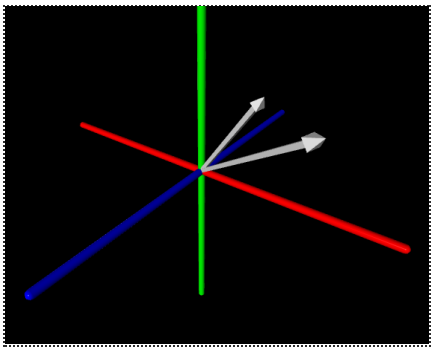


The view on the right can be rotated freely using the right-button on the mouse and dragging. The zooming actions can be performed by mouse wheel.

Let us now use another function called `arrow()` to represent the vectors `r` and `s`. The syntax for calling arrows is as follows; the tail needs to be on the origin (`pos`) and the head on the values of `r` components (`axis`).

```
r = arrow(pos=vector(0,0,0),axis=vector(5,0,0),shaftwidth=0.1)
```

```
7
8  r = vector(2.3,3,1.4)
9  s = vector(3.7,3.1,2.5)
10
11 r_v = arrow(pos=vector(0,0,0),axis=r,shaftwidth=0.1)
12 s_v = arrow(pos=vector(0,0,0),axis=s,shaftwidth=0.1)
```

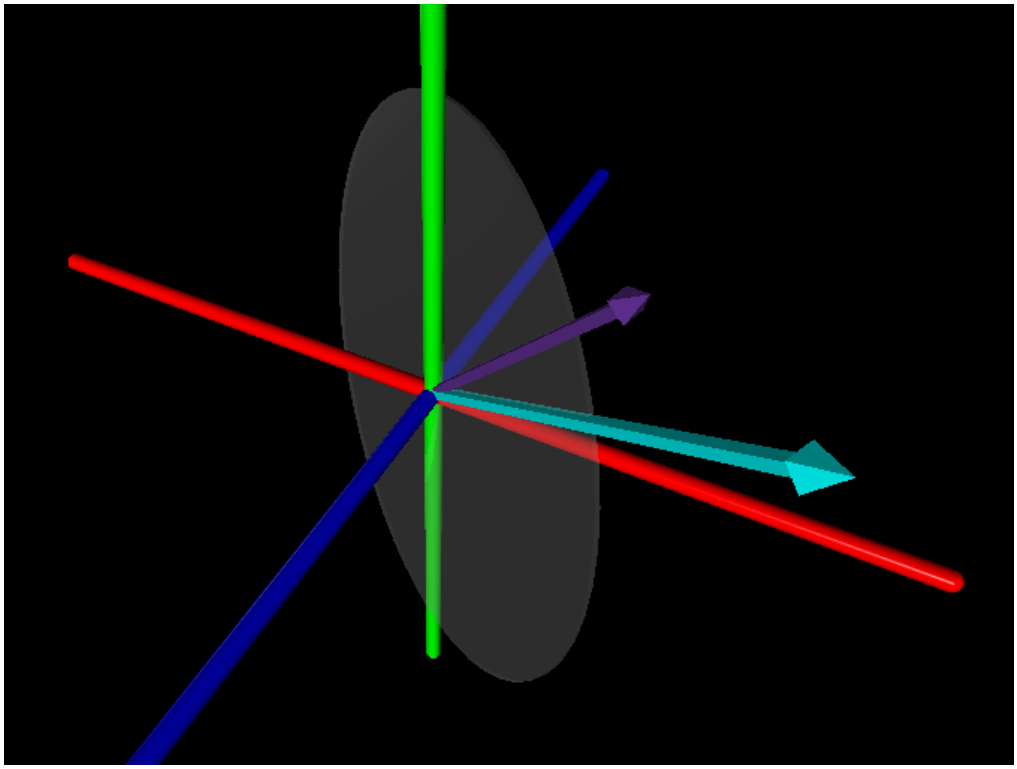


Now let us draw the `rs` plane. We do this by finding a cross product vector of `r` and `s`, and assigning it as an axis to the object function `cylinder()`. But we will first scale down the axis vector length of the cylinder to turn it into a disk. Syntax for cylinder is as follows;

`disk = cylinder(pos=vector(0,0,0),axis=vector(0.5,0,0), radius=1)`

We will use extra parameters here, such as `opacity` to help in visualization of the plane.

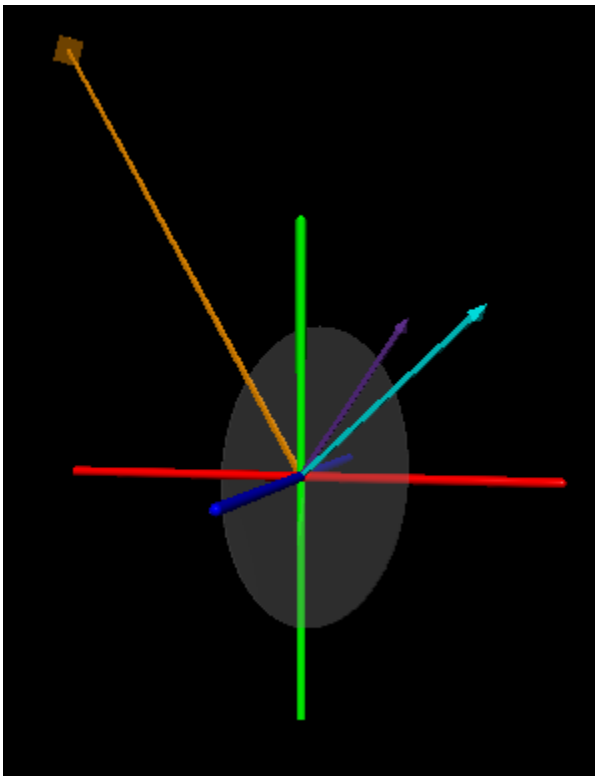
```
11 r_v = arrow(pos=vector(0,0,0),axis=r,shaftwidth=0.1, color=color.purple)
12 s_v = arrow(pos=vector(0,0,0),axis=s,shaftwidth=0.1, color=color.cyan)
13
14 rs_pvec = cross(r,s)
15 rs_plane = cylinder(pos=vector(0,0,0),axis=0.01*rs_pvec,radius=3,
16                     color=color.white,opacity=0.2)
```



You may need to rotate around and satisfy yourself that this is what a vector plane looks like.

One of the methods we calculated the vector `t` in the last exercise using the relation: $t = cross(r, cross(r, s))$. Let us draw this vector using the arrow object function as well. Surprisingly, this vector has a length because we applied cross product twice. But you can follow any of your own methods as long as the new vector lies in the rs plane.

```
17
18 t = cross(r, cross(r,s))
19 t_v = arrow(pos=vector(0,0,0),axis=t,shaftwidth=0.1, color=color.orange)
20
```



TASK ↦ Take two new vectors of your choice (try including negative components as well) and draw a plane for those.

TASK ↦ Create the planes xy, yz and xz using `box` object. [Instruction on box]

Position vectors:

In Physics, position vectors define the position of any object with respect to the coordinate axis. We have already visualized these vectors in the form of `r` and `s` in the previous section. But there was no object to point to, here, we will create objects in space and provide position vectors to represent their positions.

```
GlowScript 3.1 VPython
from visual import *

origin = vector(0,0,0)
x_axis = curve(pos=[vector(-5,0,0),vector(5,0,0)], color=color.red, radius=0.1)
y_axis = curve(pos=[vector(0,-5,0),vector(0,5,0)], color=color.green, radius=0.1)
z_axis = curve(pos=[vector(0,0,-5),vector(0,0,5)], color=color.blue, radius=0.1)

a_thing = sphere(pos=vector(-2.1,4,-1.1), radius=0.2, color=color.red)
b_thing = box(pos=4*vector.random(), color=color.cyan)
```

Notice that to place an object in space you already have its position in component form. Here, we used the object functions sphere and box and assigned a random vector to b_thing which is a box object.

> **TASK** ↦ run the for a couple of times and notice that the position vector changes along with the b_thing. Can you answer why?

Color Vectors:

The object you just created uses predefined colors from the color library in vpython, example color.red/blue/green/cyan/yellow etc. But you can run out of these colors very quickly when working with a large number of objects. You can now use the *space of colors!* The space can be accessed by using the vector function. A few examples of colors can be found here.

Here is one color associated with this color vector:
`color=vector(0.3,0.11,0.54)`

And a grid of some randomly colored boxes...

```
GlowScript 3.1 VPython
from visual import *

for i in range(10):
  for j in range(10):
    col = vector.random()
    b_thing = box(pos=vector(i,j,0), color=col)
```