The Three-Joint Lamp and Vectors

Nunnapad Toadithep
nisachol@cp.su.ac.th
Faculty of Science, Silpakorn University, Nakhon Pathom 73000
Thailand

Abstract : *This work expresses the relationship between the position of movement and its vector representation by an ordinary tool used in everday life, so that ordinary people can understand the concept of vectors easily. Force is a good example of a vector. A force that is applied to any object is composed of magnitude and direction. In physics, work occurrs if one pushes an object with some force for some distance. Forces that are applied to household tools such as doors, windows, lamps, etc, can be represented by vectors. This paper shows how to implement a vector object in VPython. An example of how to determine the best composition of the 3 limbs of a three-joint lamp by vector rotation is given.*

1. Introduction

A vector is a structure which has both magnitude and direction such as force in physics. In real life, we face with many applications of vectors. The movement of any object is the result of vector computation. There will be the question that if we travel for some amount of distance and in some direction, how far are we from home? From Newton's laws of motion, the ***net force*** experienced by an object is determined by computing the vector sum of all the individual forces acting upon that object. That is the net force is the result of adding up all the force vectors.

The three-joint lamp I found at the hotel in one of my journeys, shown in Figure 1, has three rotational joints . The objective of this discovery is to determine the appropriate position of light projecting to the working table that can make people see clearly.

2. The Rotation Matrix

Vectors can be rotated about any axis in Cartesian space. The following three rotation matrices can be used to roatate a given vector about the x-axis, y-axis and z-axis, respectively.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If we expressed the $x^{'}$, $y^{'}$, $z^{'}$ components of any vector, the matrix multiplication shown below gives the new vector, which rotates the original vector by an angle of θ radians about the x-axis.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R_x(\theta) \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

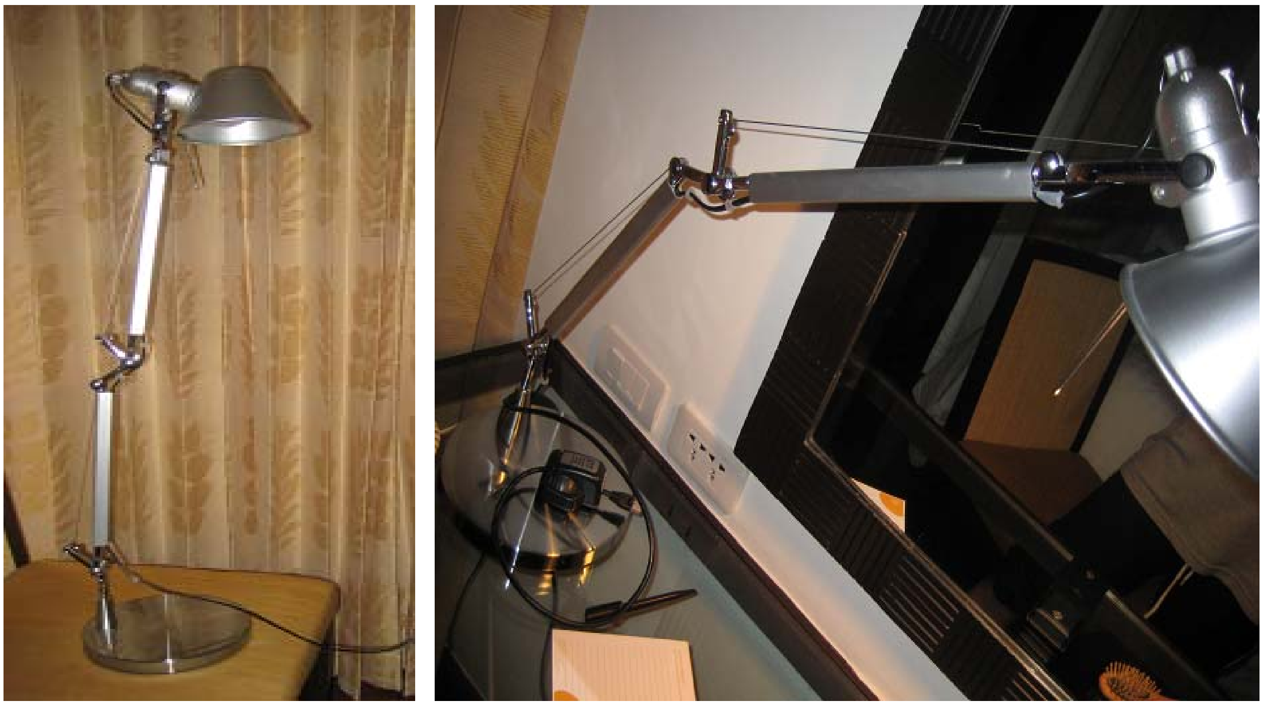


Figure 1: The Three-Joint Lamp

3. Objects in VPython

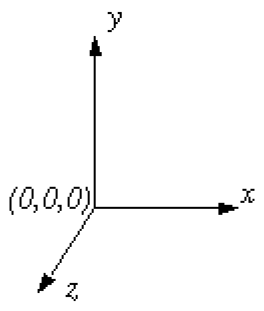
VPython is the Python programming language plus a 3D graphics module called "Visual" developed by David Scherer. This document describes all of the Visual capabilities. To invoke the Visual module, place the following statement at the start of the file (see [8]):

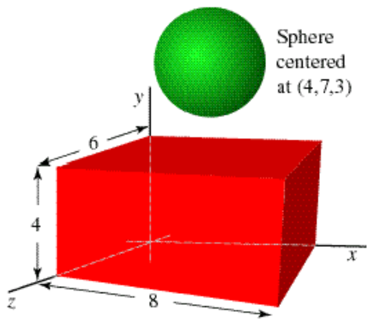
```
from visual import  *
```

VPython provides flexibility of object manipulation. With Object-Oriented Programming, objects have attributes or properties and methods to act with. There are 14 basic display objects listed below: cylinder, arrow, cone, pyramid, sphere, ring, box, ellipsoid, curve, helix, convex, label, frame and faces.

3.1 Display Window and Visual Objects

When using VPython the display window shows objects in 3D. The point (0, 0, 0) is in the center of the display window. The +x-axis runs to the right, the +y-axis runs up, and the +z-axis points out of the screen, toward you, shown in Figure 2 a):





a) Display window[9] b) Objects on the scene[9]

Figure 2: VPython window and Objects

The graphical objects such as spheres, boxes, and curves, can continue to exist for the duration of running program. The Visual 3D graphics module will continue to display the objects, wherever they are as shown in Figure 2 b). If you name the object, you can refer to that object by name such as RedBox, GreenBall. All objects have attributes or properties: there are built-in sets of attributes such as pos, color, radius, etc. If you change an attribute of an object such as its position or color, visual will automatically display the object in its new location or with its new color**.** The command below sets the position of GreenBall to (0,5,0) on the y-axis at scale 5 from the origin.

```
GreenBall.pos =(0,5,0)
```

If you want to create balls and sequence them inline, the sphere objects should be created and the radius property of objects must be invoked. For example, ball1 and ball2 have radius 0.5, so ball2 should be set next to ball1, ball1.pos must be referred to and added with 2*radius and set to be the pos property of ball2, in x-axis, the first value of tuple. We use the following two VPython commands to achieve this:

```
ball1 = sphere(pos=(1,2,1),  radius=0.5)
ball2 = sphere(pos=(ball1.pos +( 2*.5,0,0)),  radius=0.5)
```

3.2 Vector Object

A vector is a specific mathematical structure which represents the magnitude and the direction simultaneously. Wind, for example, has both a speed and a direction and, hence, can be conveniently expressed as a vector. The same can be said of moving objects and forces. The location of a point in the Cartesian space is usually expressed as a tuple (x, y,z). Being a vector, (x, y,z) has a certain distance (magnitude) and an angle (direction) relative to the origin (0, 0, 0).

In Vpython, a vector object is not a displayable object but it is a powerful aid to 3D computations. Its properties are similar to vectors used in science and engineering. There are basic manipulation of vectors which are shown below:

1. Creation of vectors from tuple
```
v1 = vector(1,2,1)
v2 = vector(3,2,0)
```

2. Addition, Subtraction and multiplication of vectors
   ```
   Addresult  = v1+v2              # equal vector (4,4,1)
   Subresult  = v1-v2              # equal vector (-2,0,1)
   Newvector = 3*v1                # equal vector (3,6,3)
   ```
3. Magnitude or length of a vector
   mag(v1)   # $\sqrt{(v1_x)^2 + (v1_y)^2 + (v1_z)^2} = \sqrt{6}$
4. Normalized vector
   Function norm() makes a unit vector of input parameter where the magnitude of unit vector is equal to 1.
   ```
   norm(v1)             # v1/ mag(v1)
   ```
5. Angle between vectors
   ```
   v1.diff_angle(v2)       # angle between vector v1 and v2
   ```
6. Dot product
   dot(v1, v2)   # dot product of vector v1 and v2= |v1||v2|cos(θ)
7. Cross product
   Creates the cross product of two vectors, which is a vector perpendicular to the plane defined by vector1 and vector2:
   ```
   cross( v1, v2 )          # cross product of vector v1 and v2
   ```
8. Rotation of a vector
   A vector can be rotated about any axis in Cartesian space, where one must specify the angle and an axis to rotate about, as given in the following example:
   ```
   v1 = v1.rotate( angle=theta, axis=(1,1,1))
   ```

   The default axis for a rotation is (0,0,1), the z-axis in the xy-plane. The value of angle indicates the direction of rotation, such that, positive value is a counterclockwise rotation, negative value is a clockwise rotation.

Note : vector (1,0,0), vector (0,1,0), and vector (0,0,1) are standard basis unit vectors.

3.3 Arrow Object

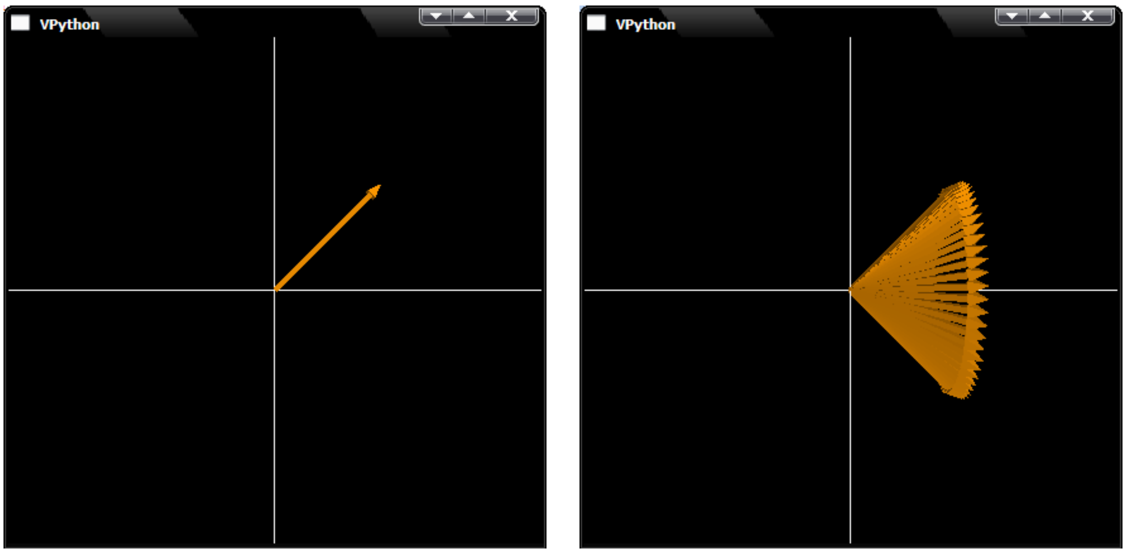
An arrow object is used to display a vector, because the vector command by itself is an invisible object. Let v be the vector (2,2,0), and to display v at the origin (0,0,0), called the position vector, one can use the arrow command. The output is a vector which is represented by an arrow, the orange one as shown in Figure 3 a):

```
v= vector(2,2,0)
arrow(pos =(0,0,0), axis = tuple(v), shaftwidth=.1, color=color.orange)
```

Let us now repeat the commands below:

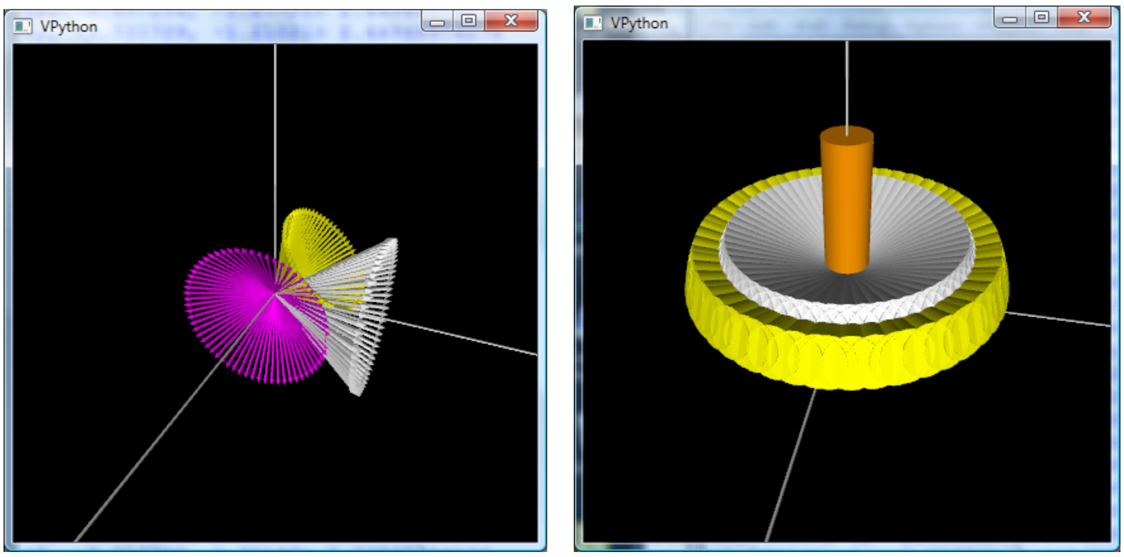
```
v=v.rotate(angle = .1, axis = (1,0,0))
arrow(pos=(0,0,0), axis= tuple(v), shaftwidth=.1, color= color.orange)
```

As a result of the above commands, we get the scene of counterclockwise rotations by .1 radians angle about the x-axis. The command v.rotate rotates a given vector, whose two attributes are the angle of rotation and the axis for rotation. The above two commands yield the result shown in Figure 3 b).



a) b)

Figure 3: Vector and its rotation about x-axis



a) b)

Figure 4: Vectors and their rotation about other axis

Given below are the sample commands to create vectors and rotate them. The result of the vector rotation which are seen in arrows, because, here we represent the arrow object as vector object, which is not displayable.

```
v1=vector(2,1,0)        # the white one
v2=vector(1,.5,0)       # the yellow one
v3=vector(1,1,1)        # the purple one
for  i  in arange(0, 2*pi , .1):
        v1=v1.rotate(angle= -.1, axis=(1,0,0))
        v2=v2.rotate(angle= -.1, axis=(1,1,0))
        v3=v3.rotate(angle= -.1, axis=(0,0,1))
```

Figure 4 a) displays the three vectors v1, v2, v3 which rotate about (1,0,0) (x-axis), (1,1,0) (xy-axis) and (0,0,1) (z-axis), respectively. Figure 4 b) is another example of vector rotation, represented by cylinder, which gives a beautiful scene.

3.4 Compare the Rotation with the Rotation Matrix

The test vector is the vector (2,2,0), as shown in Figure 3 a). In order to rotate this vector about the z-axis using the rotate command, one can use VPython as follows:

```
for  i  in range(0, 8):
     v= v.rotate(angle = pi/4, axis =(0,0,1))
     arrow(pos=(0,0,0), axis=tuple(v), color=color.yellow)
     print "vector v using .rotate  ",v
```

The output of 8 times rotation with pi/4 angle each, the displayed vectors are shown in Figure 5, with the resluting 8 vectors given right below:

```
vector v using .rotate   <2.22045e-016,  2.82843,  0>
vector v using .rotate  <-2, 2, 0>
vector v using .rotate  <-2.82843,  6.66134e-016,  0>
vector v using .rotate  <-2,  -2, 0>
vector v using .rotate  <-1.11022e-015, - 2.82843,  0>
vector v using .rotate  <2, -2, 0>
vector v using .rotate  <2.82843,  -1.77636e-015,  0>
vector v using .rotate  <2,  2, 0>
```

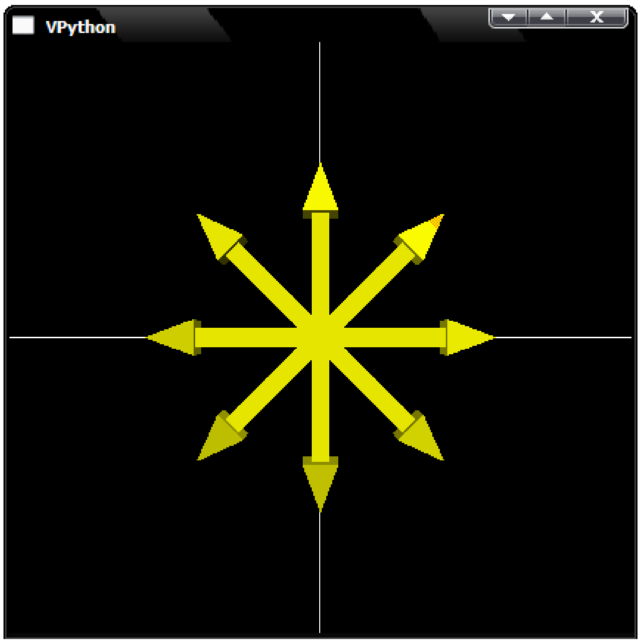


Figure 5: pi/4 radians counterclockwise rotation

The following function new_vecZ(v, r) which has 2 parameters, input vector and angle to rotate, computes matrix multiplication using VPython command dot and then converts to a vector:

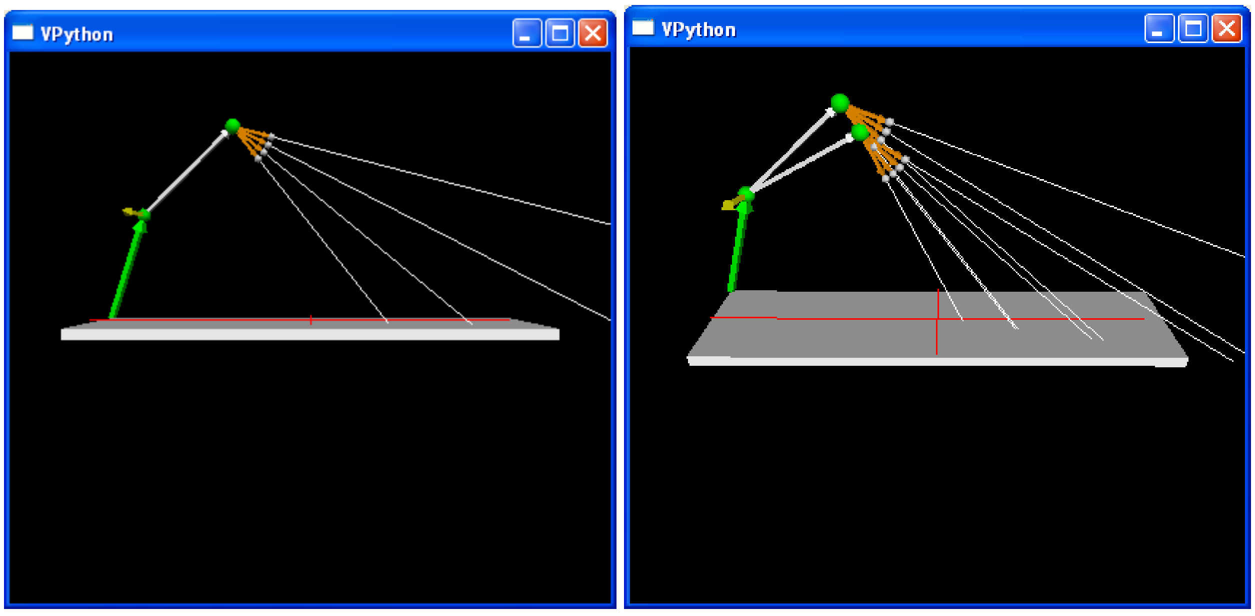
```
def  new_vecZ (v, r):
        # rotate vector v about z-axis,  r radians
        a=array ([[cos(r), -sin(r),0], [sin(r),cos(r),0], [0, 0, 1]])
        b=array ([[v.x],[v.y],[v.z]])
        c= dot (a,b)   # multiply matrix a and b
        return vector (c[0][0],c[1][0],c[2][0])
```

The following are the commands to process this job:

```
for i in  range(0, 8):
        v= new_vecZ(v, pi/4)
        arrow(pos=(0,0,0), axis=tuple(v), color=color.yellow)
        print "vector v using matrix ",v
```

The results of rotation using rotation matrix as given above are the same as the vector rotation, method rotate. In this paper, we use the built-in command rotate to rotate the limbs of the three-joint lamp:

```
vector v using matrix  <2.22045e-016,  2.82843,  0>
vector v using matrix  <-2, 2, 0>
vector v using matrix  <-2.82843,  6.66134e-016, 0>
vector v using matrix  <-2, -2, 0>
vector v using matrix  <-1.11022e-015,  -2.82843, 0>
vector v using matrix  <2, -2, 0>
vector v using matrix  <2.82843,  -1.77636e-015, 0>
vector v using matrix  <2,  2, 0>
```



a) b)

Figure 6: The 3 limbs rotation results

4. Experimental Results

This work simulates the rotation of lamp's limbs which has the end-effector as a light. The purpose of this simulation is to find the appropriate position of limbs so that one receives proper lighting at the table. We represent the first limb, the green one, by position vector (1, 6, 0.25), which has angles 80.55, 9.75, and 87.65 in degrees, respectively with the x-axis, y-axis and z-axis. The second limb is represented by the position vector (3, 6, 0). Therefore, length of the first limb is 6.09 units and length of the second limb is 6.70 units. We can use vector built-in commands to determine the angles, in degrees, of the limb with the x-axis, y-axis, and z-axis as follows:

```
dx = v1.diff_angle(vector(1,0,0))*180/pi
dy = v1.diff_angle(vector(0,1,0))*180/pi
dz = v1.diff_angle(vector(0,0,1))*180/pi
print "angle in degree",dx,dy,dz
```

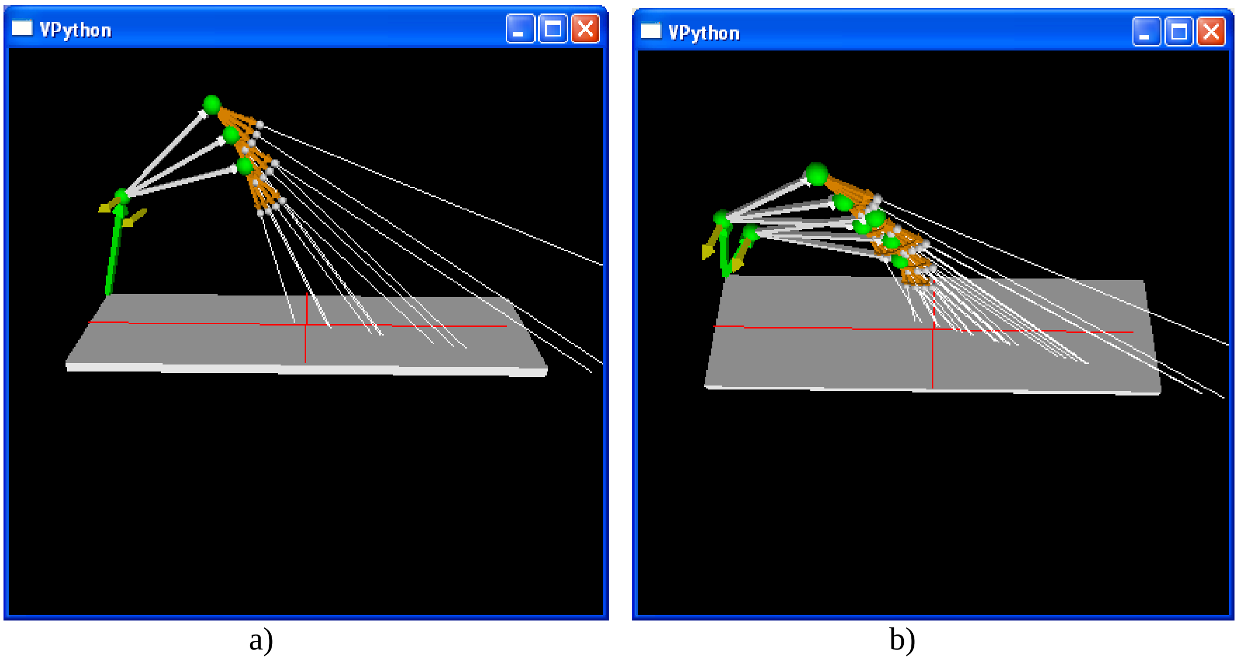


Figure 7: The results of complete rotation of the first limb

In Figure 6 a), the light scatters outside the table surface, but in Figure 6 b), the result becomes a bit better. The best result corresponds to the position vector <1.45, 0, 0.10> which is generated from the rotation of the 4th of 2nd of 1st limb. The situation where the light is incident at the origin of the table, is claimed to be the best position of light for ideal working conditions. The Figure 7 b) shows the complete work of the simulation in top view. The Figure 8 shows the best result, in which the position of the arrow-head corresponds to the second limb: the origin of light is at position <-3.15, 9.15, -1.04> on a table of size (22, 0.5, 6), i.e. 22 in length, 0.5 in height, and 6 in width.

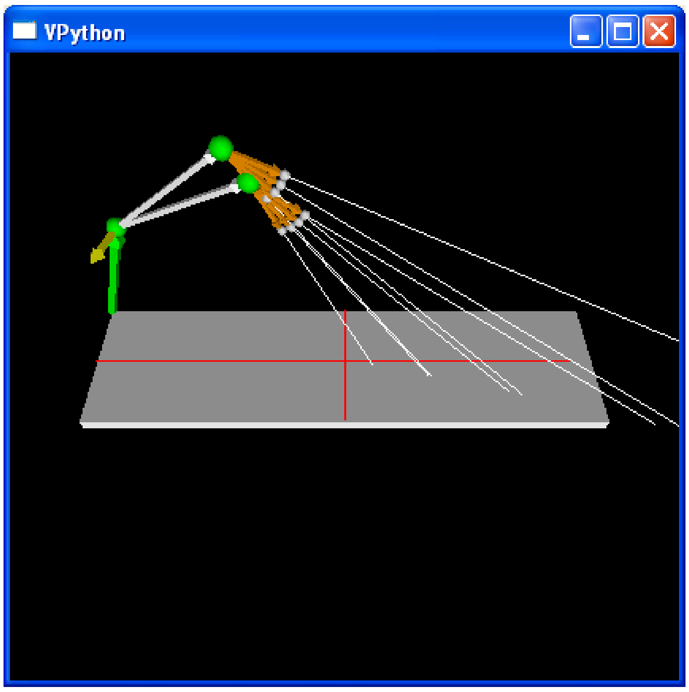


Figure 8: The leftmost light, the best position

Because the second limb has to rotate about the axis of a vector that is perpendicular to the first limb, this vector is created as the cross product of the first limb vector and its very close vector. This cross product is a vector, a unit vector, used as a base vector, called vector C, and can be seen in yellow of Figures 6 to 8. This base vector plays an important role in our simulation.

4.1 Algorithm of Simulation

```
1.  repeating the rotation of the first limb vector

    1.1  rotate 2 times, with the angle -.2 radians each, about the axis of base vector, vector C.
    1.2  repeating the rotation of the second limb vector
         1.2.1  rotate 3 times about the base axis (vector C), with the angle -.2 radians each
         1.2.2  repeating the rotation of the third  limb vector
                1.2.2.1  rotate 4 times about the base axis (vector C), with the angle -.2 radians
                         each
                1.2.2.2  display the position of  light that goes toward the table
```

5. Conclusion

In real life, there are many things to be discovered and many of them are related to mathematics. We face vectors in all our daily routines, such as driving a car, walking, or eating, or doing any other work. You hardly ever stay clear of vectors, as every person lives under mathematics, or mathematics covers the world. So everyone should desire, and be happy to discover things in one's next journey.

References

[1] “Learning Inverse Kinematics”. Proceeding of the 2001 IEEE/RSJ International Conference on Intelligent Robot and Systems, Maui, Hawaii, USA, Oct. 29- Nov. 03, 2001.
[2] “Real-time Inverse Kinematics Techniques for Anthropomorphic Limbs”. Graphical Models 62, 353-388(2000).
[3] “Implementation of Inverse Kinamatics and Application”. http://cs.virginia.edu/~gfx/courses/2000/...spring.../lecture10.ppt
[4] “Vector Math for 3D Computer Graphics”(2003). http://chortle.ccsu.edu/vector-lesson/vector/index.html
[5] “Vector Product Application”. http://cnx.org/content/m14522/lastest/
[6] “Vectors”. http://www.physics.vougue/ph.ca/tutorials/vectors/vectors.html
[7] “Vectors:Motion and Forces in Two dimensions”. http://www.physicsclassroom.com/class/vectors/
[8] “The Visual Module of VPython”. http://www.vpython.org/webdoc/index.html