Visualization of the electric field of a point charge[1]

Overview

In the first three laboratory activities, you will visualize and simulate phenomena that you have seen in your introductory mechanics and electromagnetism courses. Because the physics is familiar, you will able to focus on learning and/or brushing up on the Python programming language. Further, each exercise is designed to illustrate one or more important points about computational simulations that will be relevant when you move on to programming for quantum computers.

For these activities, you will use the VPython[2] programming language, which is comprised of Python plus a graphics module that makes it easy to produce real-time 3D animations. You will write and run your programs in the GlowScript[3] interactive development environment (IDE), which runs in your web browser. The GlowScript IDE cannot access your computer's file system or Python's add-on packages. Some common functions from the *math* and *numpy* modules are, however, incorporated into GlowScript VPython.

Objectives

In this laboratory, you will write code to calculate and visualize the electric field $\vec{E}$ at many locations around a single point charge. Clearly, once the number of locations increases beyond a few, it becomes impractical to calculate the value of $\vec{E}$ using pencil and paper, even with the aid of a calculator. This task is perfect, however, for a computer because ***computers excel at performing simple repetitive numerical calculations***.

More generally, there are numerous phenomena throughout physics and other scientific disciplines that can only be studied using computers. These areas of study fall under the umbrella category of **scientific computing**; Quantum Information Science is one such topic.

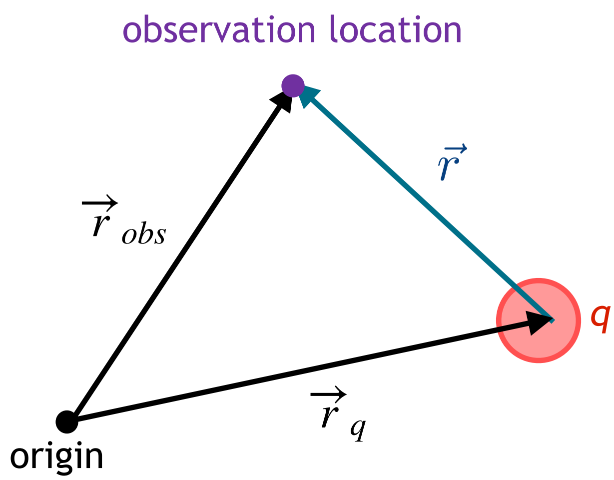
After completing this activity, you should be able to:

- *Create, fill, and loop through a Python list.*
- *Define and use your own Python function.*

[1] Adapted from *VPython Introductory Computational Physics* by Ruth Chabay and Bruce Sherwood.

[2] VPython was developed by Carnegie-Mellon physicists David Scherrer and Bruce Sherwood.

[3] VPython is also available for installation on individual computers as a Python module. See vpython.org for more information.



Electric field of a point charge

A point charge q electric field produces an electric field $\vec{E}$ at all points in space. The magnitude and direction of $\vec{E}$ are given by **Coloumb's Law**:

$$\vec{E} = K \frac{q}{r^2} \hat{r} \,,$$

where

K = 8.99 x 10^9 N m^2/C^2 (the electrostatic constant, also known as the Coulomb constant)

r = the distance between the charge *q* and the *observation location*

$\hat{r}$ = a unit vector pointing *from* the charge *q* *toward* the observation location.

Given the position of the source charge, $\vec{r}_q$, and the location at which we want to know the electric field, $\vec{r}_{obs}$, it is easy to find r and $\hat{r}$. As shown in the figure at at the top of the page, the position of the observation location relative to charge q is given by the vector pointing to the observation point minus the vector pointing to the source charge:

$$\vec{r} = \vec{r}_{obs} - \vec{r}_q \,.$$

1. Getting started with VPython in GlowScript

- Sign in at www.glowscript.org. (If necessary, create an account.)
- You will see the sentence "You are signed in as <your user name> and your programs are **here**." Click on **here**.
- Using the **Add Folder** tab, create a folder named **QCIPU**. *Make sure that Public is checked* when you do so — this will allow your instructors and peers to see your programs.

```
GlowScript 3.1 VPython
## Electric field of Point Charge
## Ruth Van de Water, ruthv@fnal.gov
## last edited: July 5, 2021

scene.width = scene.height = 550
scene.caption= """To rotate scene, drag with right button or Ctrl-drag.
To zoom, drag with middle button or Alt/Option-drag or scroll wheel.
To pan left/right and up/down, Shift-drag."""

## constants
global K
K = 8.99e9     #Coulomb constant in SI units
q_e = 1.6e-19 #electron charge in Coulombs
scalefactor = 1e-20 #needed to make E-field arrows a reasonable size

## shpere representing charge qe located at the origin
r_q = vec(0,0,0)
charge = sphere(pos=r_q, radius=5e-11,color=color.magenta)

## E-field at position r_obs due to charge q
r_obs = vec(-3e-10,0,0)
r = r_obs - r_q
E = K * q_e * hat(r)/mag(r)**2.

## arrow at r_obs whose direction is parallel to the E-field vector
## and whose length is proprtonal to the magnitude of E
arrow(pos=r, axis=scalefactor * E, color=color.yellow)
```

2. Electric field at a single location

- Go into the QCIPU folder and select **Create New Program**. Name your program *Lab1* or something similar. You will then be sent immediately to the GlowScript editor for this program.
- Copy the above code into your GlowScript editor, but **do *not* run it**. Spend *at least 5 minutes* going through the code with your partner(s) making sure that you understand what it does at every single line before proceeding to the next step.
- Click **Run this program**. *Does the code do what you expected? Are the magnitude and direction of* $\vec{E}$ *physically reasonable? If not, why not? Be sure to fix any errors and/or resolve any discrepancies before proceeding to step 3.*

3. Electric field at multiple locations

For the remainder of this activity, you will modify the provided starting code so that it calculates and draws arrows representing the electric field at *many* locations surrounding the point charge. As with any programming task, the best approach is to break it into several steps, each of which can be debugged and checked before proceeding to the next one.

3a. Calculation of the electric field

Whenever you need to perform same operation more than once or twice in a program, it is most efficient to write a **function**. ***Functions keep your code short and readable, and also prevent errors from being introduced when copying-and-pasting.***

- Based on the code in lines 23—28, write a function that calculates the electric field vector at $\vec{r}_{obs}$ due to a point charge at $\vec{r}_q$ and then draws an arrow representing the electric field at that location. *What parameters should your function depend on? What value(s) should it return?*
- ***Pause and check!*** Your function should (*for identical inputs*) obtain the same electric field as lines 23—24 and produce the same arrow as line 28 in the starting code.
- Once you are confident that your function is working correctly, move it to an appropriate location near the beginning of your program. Then replace lines 23—28 with a single call to your new function.

Check in with an instructor before continuing.

3b. Electric field in the *xy* plane

Whenever you need to perform the same operation multiple times using different values for the input parameters, it is useful to put the different values for each parameter in a **list or other iterable container**. Once your parameter values are in a list, ***you can loop over the list elements and perform the desired operation(s) on each element one-at-a-time with only a few lines of code***.

The electric field of a point charge is *spherically symmetric*, meaning that it looks the same viewed from any angle as long as you remain at a fixed distance from the charge.

- Create a list of vectors at 12 evenly spaced locations on a circle of radius $R = 3 \times 10^{-10}$ m, lying in the *xy* plane and centered on the charge.

  Hint: Although VPython vectors are given in *Cartesian (x,y,z) coordinates*, a circle is best described in *spherical polar* (r, θ, ϕ) *coordinates*. *How do you express the spherical-polar vector* (r, θ, ϕ) *in Cartesian coordinates? What is the value of* ϕ *in the xy plane?*
- Replace the single location vector $\vec{r}_{obs}$ in line 22 of the starting code with your new list. Then write a **loop** that calculates and draws the $\vec{E}$-field at each location in the list. *Remember to use the function that you wrote in 3a.*
  - Syntax for a **for** loop: https://www.w3schools.com/python/python_for_loops.asp
  - and for a **while** loop: https://www.w3schools.com/python/python_while_loops.asp
- ***Pause and check!*** *Are your arrows evenly spaced around the point charge? Is the electric field spherically symmetric? (You may need to rotate the scene to get a good view.)* If not, fix your code as needed.

The *magnitude* $|\vec{E}|$ of the electric field from a point charge decreases with the distance from the charge as $1/r^2$.

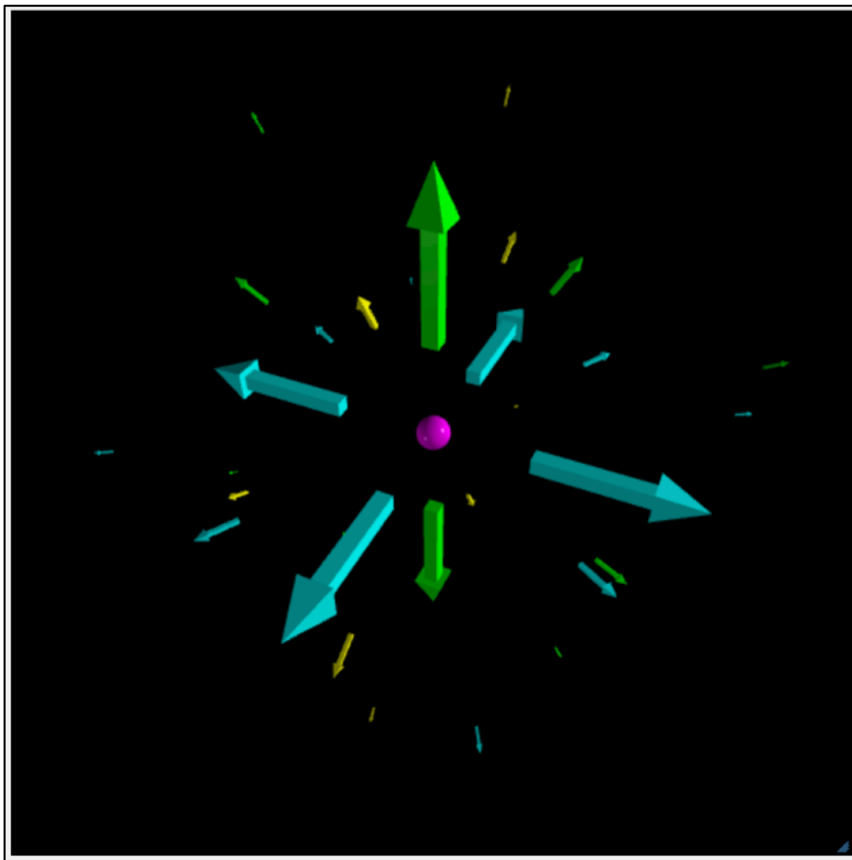
- Modify your code to also calculate and draw the electric field at 12 evenly spaced locations on circles in the *xy* plane of radii $R = 6 \times 10^{-10}$ m and $R = 9 \times 10^{-10}$ m
  Hint: You can do this quickly by creating a new list **R_list = [3e-10, 6e-10, 9e-10],** and looping over R_list.
- ***Pause and check!*** *Does the magnitude of the electric field decrease with the distance from the charge as* $1/r^2$*? (You may need to rotate the scene to get a good view.)* If not, fix your code as needed.

Check in with an instructor before continuing.

3c. Electric field in the *xz* and *yz* planes (*if time allows*)

- Repeat step 3b with circles in the xz and yz planes.
- ***Pause and check!*** *Rotate the scene around to get a good view from many angles. Do the magnitude and direction of the electric field look physically reasonable?*

Just for inspiration… your final product should look something like this. Pretty, right? ;)



To rotate scene, drag with right button or Ctrl-drag.
To zoom, drag with middle button or Alt/Option-drag or scroll wheel.
To pan left/right and up/down, Shift-drag.