Abhishek Sebastian 20BEC1118

Electromagnetic Field Theory - Python UI Application Assignment .

Application Introduction:

This Application is Built Using PyQt Platform, and Python Language. This application is primarily is used to Calculate The Electric field of a Point Charge.

Application Complexity:

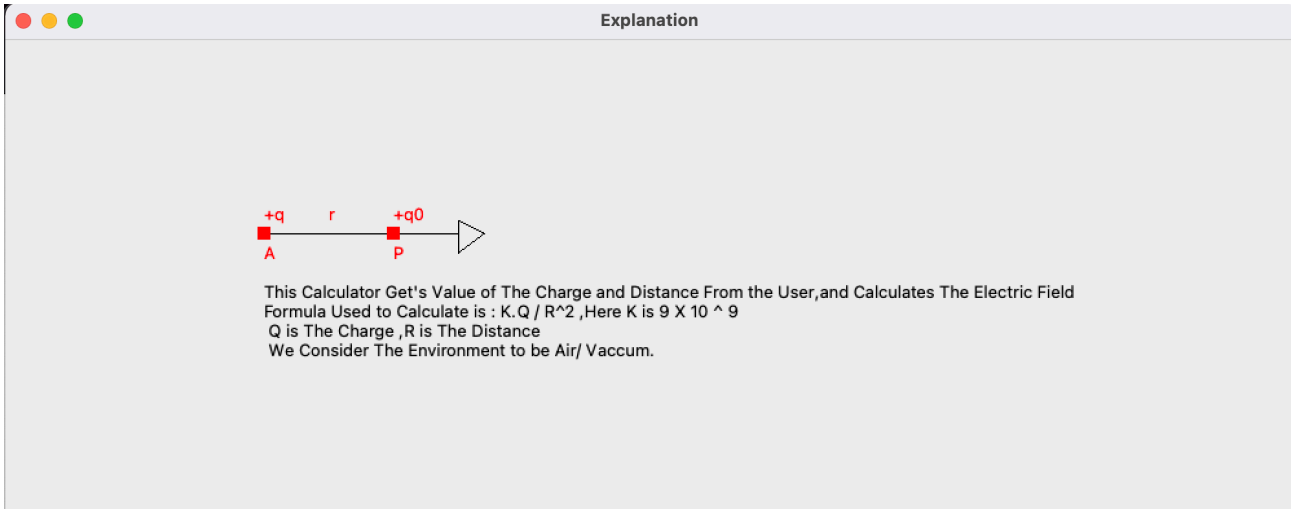
This Application obtains the Value of The Point Charge(C), and The Value of Distance(M) from The User and Provides The Electric field in terms of (N/C).

Application Demonstration:

1) To Be deployed Using the Terminal Window.

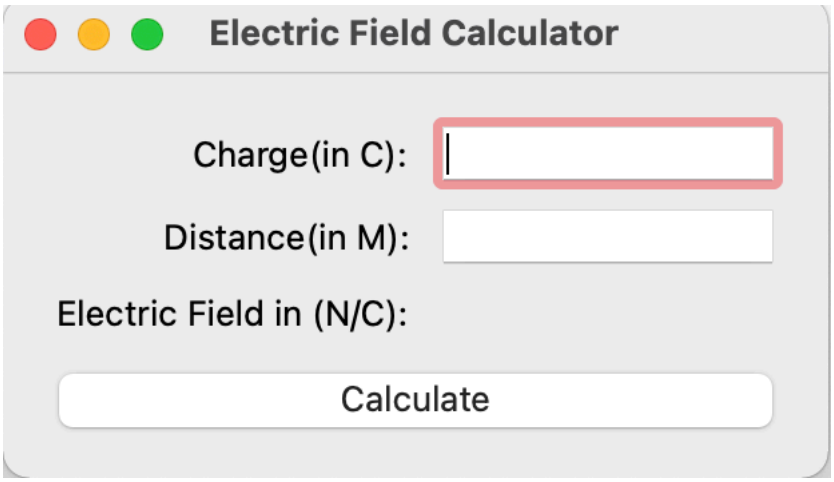
```
(base) abhisheksebastian@abhisheks-mbp App % python addd.py
```

2)

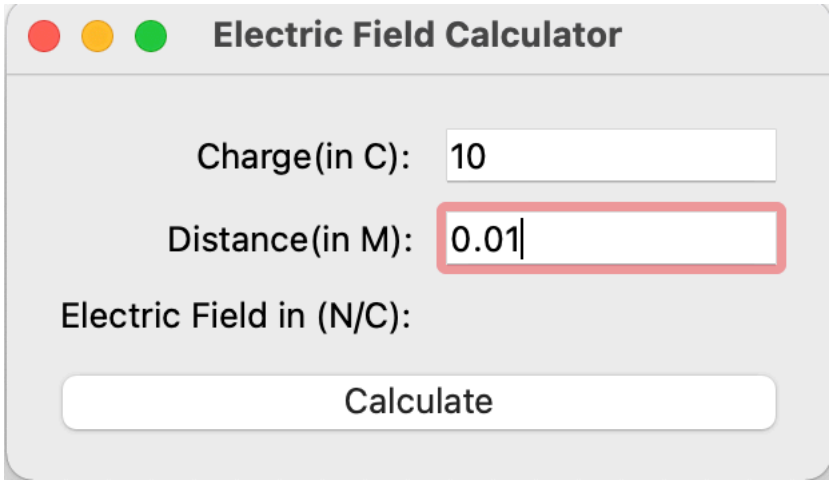


Explanation Window is Popped , along with Calculator Widget.

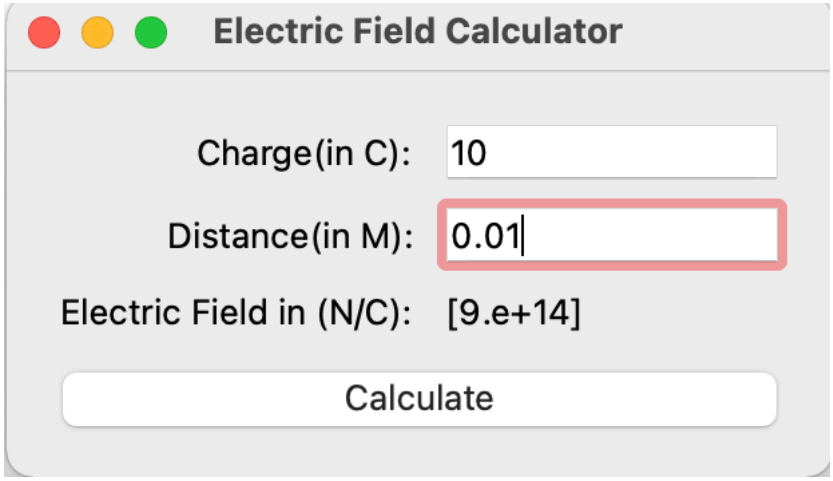
3)



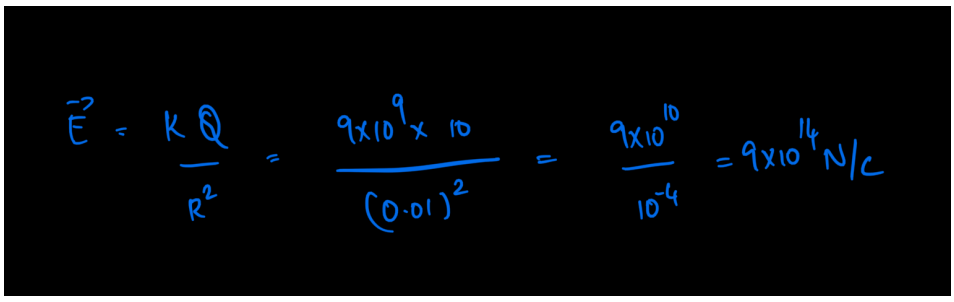
4) Enter Values Of The Charge and Distance in The Edit Text rows.



5) Press Calculate.



We Get The Electric Field to be found at 9. X 10 ^14 , Which is Theoretically Proved as below.



Code:

```
from PyQt5 import QtCore, QtGui, QtWidgets
from math import prod
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QPixmap
import numpy as np
from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton, QLabel, QVBoxLayout, QWidget

from numpy import multiply, product

class Widget(QtWidgets.QWidget):
    def __init__(self, parent=None):
        super(Widget, self).__init__(parent)
        flay = QtWidgets.QFormLayout(self)

        regex = r"^(\s*(-|\+)?\d+(?:\.\d+)?\s*,\s*)+(-|\+)?\d+(?:\.\d+)?\s*$"
        validator = QtGui.QRegExpValidator(QtCore.QRegExp(regex), self)
        self._le = QtWidgets.QLineEdit()
        self._le_1 = QtWidgets.QLineEdit()


        self._le.setValidator(validator)
        self._le_1.setValidator(validator)
        self.setWindowTitle("Electric Field Calculator")

        self._list_widget = QtWidgets.QListWidget()

        button = QtWidgets.QPushButton("Calculate")
        button.clicked.connect(self.on_clicked)

        self._result_label = QtWidgets.QLabel(alignment=QtCore.Qt.AlignCenter)

        flay.addRow("Charge(in C): ", self._le)
        flay.addRow("Distance(in M): ", self._le_1)
        flay.addRow("Electric Field in (N/C): ", self._result_label)
        flay.addRow(button)

    @QtCore.pyqtSlot()
    def on_clicked(self):
        self._list_widget.clear()
        if self._le.text() and self._le_1.text() :
            charge_values = [float(C_val) for C_val in self._le.text().split(",")]

            distance_values = [float(D_val) for D_val in self._le_1.text().split(",")]


self._result_label.setText(str(np.divide(multiply(charge_values,9e9),multiply(distance_values,distan
ce_values))))
            Final=
str(np.divide(multiply(charge_values,9e9),multiply(distance_values,distance_values)))


            """self.setGeometry(0, 0, 400, 300)
            self.label = QLabel(self)

        # loading image
            self.pixmap = QPixmap('image.png')

        # adding image to label
            self.label.setPixmap(self.pixmap)

        # Optional, resize label to image size
            self.label.resize(self.pixmap.width(),self.pixmap.height())
            self.show()"""
class MainWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super().__init__()
        self.acceptDrops()

        # set the title
        self.setWindowTitle("Explanation ")

        # setting  the geometry of window
        self.setGeometry(200, 100, 700, 700)
```

```
        # creating label


        self.label = QtWidgets.QLabel()
        canvas = QtGui.QPixmap(1000, 1000)
        self.label.setPixmap(canvas)
        self.setCentralWidget(self.label)
        self.draw_something()


    def draw_something(self):
        painter = QtGui.QPainter(self.label.pixmap())
        pen = QtGui.QPen()
        painter.drawText(200,200, "This Calculator Get's Value of The Charge and Distance From the
User, and Calculates The Electric Field")
        painter.drawText(200,215, "Formula Used to Calculate is : K.Q / R^2 , Here K is 9 X 10 ^ 9
")
        painter.drawText(200,230, " Q is The Charge , R is The Distance ")
        painter.drawText(200,245, " We Consider The Environment to be Air/ Vaccum.")



        painter.drawLine(200, 150, 350, 150)
        painter.drawLine(350, 140,350,165)
        painter.drawLine(350, 165,370,150)
        painter.drawLine(350, 140,370,150)

        pen.setWidth(10)
        pen.setColor(QtGui.QColor('red'))
        painter.setPen(pen)
        painter.drawPoint(200, 150)
        painter.drawPoint(300, 150)

        painter.drawText(200, 140, '+q')
        painter.drawText(200, 170, 'A')
        painter.drawText(300, 140, '+q0')
        painter.drawText(300, 170, 'P')
        painter.drawText(250, 140, 'r')







        painter.end()

if __name__ == '__main__':
    import sys
    app = QtWidgets.QApplication(sys.argv)
    w = Widget()
    m=MainWindow()
    m.show()
    w.show()
    sys.exit(app.exec_())
```

Conclusion:
This Calculator Can Be Even More Improvised to Take Vectors as Inputs, and also includes SuperPosition theorem.