```
from tkinter import*
from math import*

def mouse_apertado(evento): #função que localiza a partícula quando clicada.
    global carg,booSelec,carSelec,xa,ya
    xa,ya=evento.x,evento.y
    encl=anim.find_enclosed(xa-40,ya-40,xa+40,ya+40) #localiza a partícula
    for e in encl:                          #mais próxima do cursor.
        if e in carg:
            booSelec=True
            carSelec=anim.gettags(e)[0]

def mouse_solto(evento):
    global booSelec
    booSelec=False

def mouse_movido(evento): #permite 'arrastar' a partícula
    global booSelec,carSelec,xa,ya
    if booSelec:
        xe,ye=evento.x,evento.y
        dx,dy=xe-xa,ye-ya
        pC[int(carSelec)][0]+=dx
        pC[int(carSelec)][1]+=dy
        desenhar()
        xa,ya=xe,ye

def desenhar(): #criação das cargas após digitar os novos valores de q
    global pC,rC,rc,q,carg,npts
    anim.delete(ALL)
```

```
    carg[0]=anim.create_oval(pC[1][0]-rc,pC[1][1]-rc,pC[1][0]+rc,
          pC[1][1]+rc,tags=str(1),fill="lightgreen")
    carg[1]=anim.create_oval(pC[0][0]-rc,pC[0][1]-rc,pC[0][0]+rc,
          pC[0][1]+rc,tags=str(0),fill="green")

    grd=range(1,npts)
    lq=larg/npts #lado de cada quadradinho que contém o círculo
    for i in grd:
       x=(i+1/2)*lq
       for j in grd:
          y=(j+1/2)*lq
          mostrar=True
          f=0.
          for n in range(0,2):
             xr,yr=x-pC[n][0],y-pC[n][1]
             r=sqrt(xr**2+yr**2)
             if r<rc+1.7:
                mostrar=False
                break
             f+=k*q[n]/r
          if mostrar:
             anim.create_rectangle(x-lq/2,y-lq/2,x+lq/2,y+lq/2,outline=cor(f),fill=cor(f))##
             ## cria a construção do potencial elétrico.

def cor(f):       #função que imprime cor, conforme o valor do potencial,
                  #logaritmizado a fim de
    esc=10**(-7.54)#facilitar a impressão.
    if f>0:
       lf=(f)*esc
```

```
if lf>=8:
    return "#f00"
elif lf>=7:
    return "#e00"
elif lf>=6:
    return "#d00"
elif lf>=5:
    return "#c00"
elif lf>=4:
    return "#b00"
elif lf>=3:
    return "#a00"
elif lf>=2:
    return "#900"
elif lf>=1:
    return "#800"
elif lf>=0:
    return "#700"
elif lf>=-1:
    return "#600"
elif lf>=-2:
    return "#500"
elif lf>=-3:
    return "#400"
elif lf>=-4:
    return "#300"
elif lf>=-5:
    return "#200"
elif lf>=-6:
```

```
        return "#100"
    else:
        return "#000"

elif f<0:
    lf=(-f)*esc
    if lf>=8:
        return "#0ff"
    elif lf>=7:
        return "#0ee"
    elif lf>=6:
        return "#0dd"
    elif lf>=5:
        return "#0cc"
    elif lf>=4:
        return "#0bb"
    elif lf>=3:
        return "#0aa"
    elif lf>=2:
        return "#099"
    elif lf>=1:
        return "#088"
    elif lf>=0:
        return "#077"
    elif lf>=-1:
        return "#066"
    elif lf>=-2:
        return "#055"
    elif lf>=-3:
```

```
            return "#044"
        elif lf>=-4:
            return "#033"
        elif lf>=-5:
            return "#022"
        elif lf>=-6:
            return "#011"
        else:
            return "#000"

    else:
        return "#000"


def novascargas (evento): #função que recolhe os dados digitados.
    global q
    for i in range (0,2):
        q[i]=float(qEntr[i].get())
    desenhar ()

pC=[[300.,300.],[400.,300.]]
rc=5
q=[1,1]
carg=[0,0]
larg=700
npts=250 ##<<----PARA MAIS AGILIDADE, DIMINUIR ESTE VALOR!!!!!!!
k=8.99*10**9 #N*(m²*)C^(-2), constante dielétrica do vácuo.
booSelec=False
jan=Tk()
```

```
jan.title('Potencial Elétrico gerado por duas cargas puntiformes')
anim=Canvas(jan,width=larg,height=600,bg="black")
anim.grid(column=1,row=1,columnspan=4)
anim.bind("<Button-1>",mouse_apertado)
anim.bind("<Button1-Motion>",mouse_movido)
anim.bind("<Button1-ButtonRelease>",mouse_solto)
qRot=[]
qEntr=[]
qVar=[]
for i in range(0,2):
    qRot.append(Label(jan, text="Carga " + str(i+1)+ ":"))
    qRot[i].grid(column=1+2*i, row=2, sticky=E)
    qVar.append(StringVar ())
    qVar[i].set(str (q[i]))
    qEntr.append(Entry(jan, textvariable=qVar [i]))
    qEntr[i].grid(column=2*(1+i), row=2, sticky=W)
    qEntr[i].bind("<Return>", novascargas)
desenhar()
jan.mainloop()

## O programa ficou um pouco lento, posto que foi necessário aumentar o
## número de pontos da janela (npts) a fim de melhorar os pontos de con-
## torno entre as cargas, e na interseção dos potenciais (mais visíveis
## em situações de cargas opostas). Isto foi feito apenas para fins de es-
## tética, deixando um visual mais elegante.
```