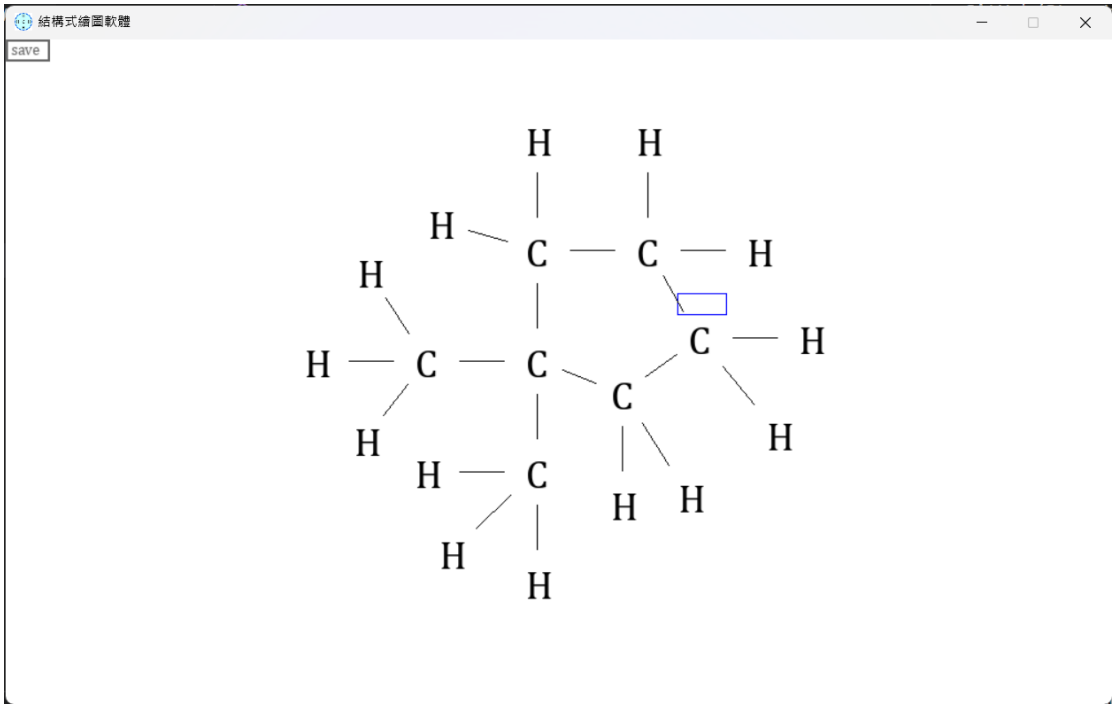
結構式繪圖軟體

目錄

摘要

此次專案同樣使用 Pygame 套件製作。因為第五冊選修化學中的主要內容是有機化學，我發現我們老師製作的講義中，圖片的解析度不佳，經過詢問得知老師是從有機課本中尋找圖片，而許多免費的繪圖軟體不符合老師的需求。於是我決定自行製作此軟體。過程中使用許多平面向量的觀念，從元素開始，再到鍵結，克服許多困難。最終成品截圖如下，在實作成果中將有近一步的說明。



我很享受這解決問題的過程，經由這個專案我又更加熟悉 Python 這個程式語言以及 Pygame 套件，同時也接觸到更多與圖像處理有關的套件，例如 PIL 使我可以將 Pygame 輸出的 RBGA Bytes 變數，用 PNG 的形式存到使用者的設備中。

心得

此專案耗費我 12 小時整的心力(不包含撰寫此報告的時間)，我再次應用我高中所學的數學與程式設計能力。我在高中參加過北一區資訊學科能力競賽，即使此專案沒有應用到所有我學過的演算法與資料結構，但經過 C++與思考訓練，我變得可以在短時間內學習一種新的程式語言或該語言中的衍生套件。

起初我是為了開發遊戲以外的應用程式，以探索新的應用領域。我發現不同類型的應用程式有不同的困難點，此外，程式語言的特性也會影響開發難度。

困難

1. Python 的物件在複製時會被當作參考(Reference in Cpp)，雖然有時候這樣對我來說很方便，但是大多數情況下我希望他是被 Deep Copy，也就是在修改時不會影響原來的變數值。
2. 在處理存檔問題時，我始終沒有找到讓使用者選擇檔案路徑的完美方法。我使用 tkinter 中的 filedialog，然而這樣使用者必須點擊兩次。第一次選擇檔案建立的位置，第二次再次選擇該檔案使檔案確實寫入資料夾。
3. 同樣是在存檔時，我總是存到空白的圖片。隨後發現是因為我在每個畫面渲染之前存檔所致。

製作動機

我希望能用我程式設計的能力幫助我的教師，那些曾經幫助過我的人。這個軟體幫助的是化學老師。因為發現到他對畫出結構式的需求，而決定製作此軟體。

製作過程

1. 物件 Vec2D 是從我的 Pygame Diep 專案複製過來的。裡面包含許多二維向量的運算，以及轉換成 Python 裡面的 tuple 的方法(method)或稱為成員函式(member function)。
2. 我首先製作有機化學分子式的核心：元素(Element)。元素物件所需要的變數如下。
   (1) 元素符號(字串)
   (2) 所在位置(Vec2D)
   (3) 是否被選擇(bool)
   (4) 唯一代碼(int)，詳見未來展望
3. 而元素需要能夠被點擊。因此需要一個函式偵測滑鼠的位置是否再元素的可點擊區域中。而主程式中就需要對每一個元素呼叫這個函式，並接受回傳值。回傳值表示滑鼠點擊符合的判定種類，以此決定接下來的操作，包含新增元素在上下左右，移動/刪除元素，以及選取元素。(此函式為 detect_mouse)

4. 再來是同樣重要的鍵結，鍵結需要的變數如下。

   (1) 起始元素(Element)

   (2) 結束元素(Element)

   (3) 單/雙/三鍵(int)

   (4) 唯一代碼(int)

5. 顯示元素與鍵結，我使用 pygame.font 的函式 font.render 顯示元素，用 pygame.draw.line 繪製鍵結。而雙/三鍵的位置用到法向量調整位置。

6. 鍵結需要能夠被選取，然而鍵結是一條線，因此我決定使用直線距離與向量內積作為鍵結的偵測標準。

   說明：假設元素 AB 之間有一個鍵結，設 $A(a_1,a_2)$，$B(b_1,b_2)$。則可以構造一個 AB 的直線方程式：

   $$(a_2-b_2)x-(a_1-b_1)y-\big(a_1(a_2-b_2)-a_2(a_1-b_1)\big)=0$$

   設$f_1=(a_2-b_2), f_2=-(a_1-b_1),$

   $f_3=\big(a_1(a_2-b_2)-a_2(a_1-b_1)\big)$，點擊的位置為$P(p_1,p_2)$

   則 P 到 AB 的距離為$\dfrac{|p_1f_1+p_2f_2+f_3|}{\sqrt{f_1^2+f_2^2}}$

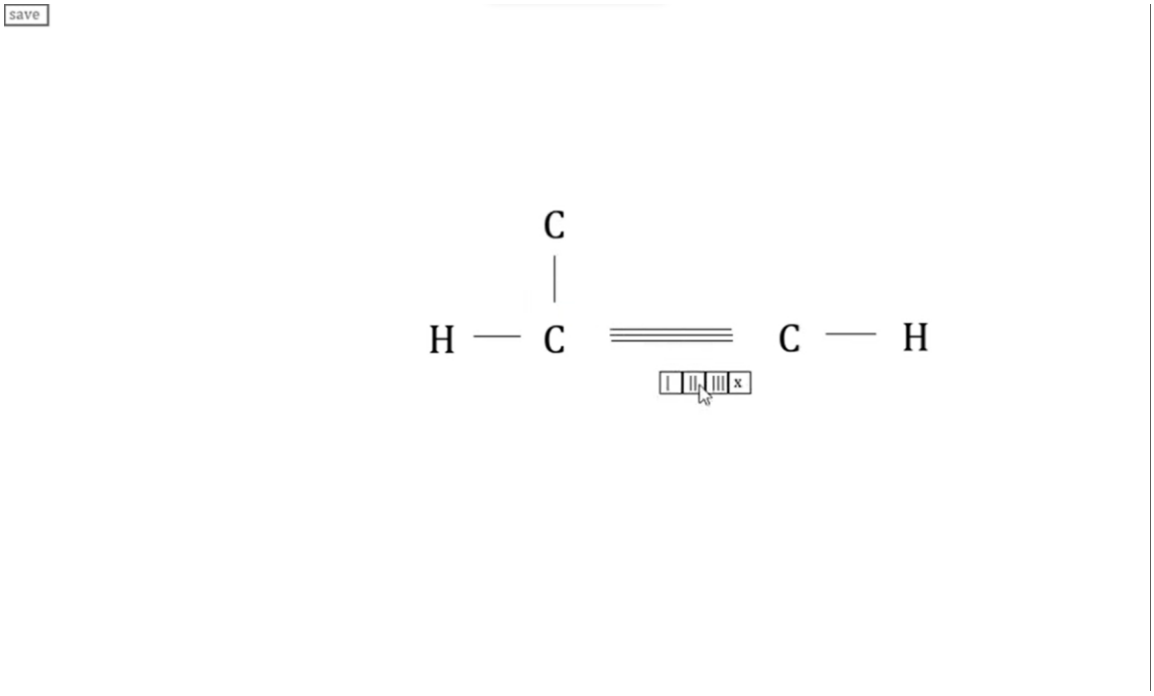
   而若向量$PA\cdot PB\leq 0$，則 P 點必定落在以 AB 中點為圓心，半徑為$\frac{1}{2}AB$的圓。如此可以確保滑鼠點擊的位置在這兩個區域的交集上。

7. 點擊鍵結時，在下方顯示出四個按鈕，分別表示要將它更改為單雙三鍵，此時會用到鍵結的唯一代碼，表示這四個按鈕屬於哪一個鍵結。按下按鍵後，依據按鍵的文字(|, ||, |||, x)更改鍵結數目。如果按下 x 則刪除鍵結。
8. 移動元素：我在使用者點擊元素後，儲存玩家點擊的位置，若玩家沒有放開滑鼠，則將元素的座標向滑鼠移動。

實作成果

成果影片連結：https://youtu.be/9MscGID9zAg

使用過程中的截圖如下。



主程式程式碼如下。註解有部分是測試時留下的程式碼。若有需要，點此跳到未來展望。

```
import pygame
from tkinter import filedialog as fd
import Element
import os
import zlib
from PIL import Image

# pygame init
pygame.init()
screen=pygame.display.set_mode((1000,600))
screen.fill((255,255,255))
pygame.display.set_caption("結構式繪圖軟體")
font=pygame.font.SysFont('cambriamath',36)
clock=pygame.time.Clock()
programIcon=pygame.image.load('icon.jpg')
pygame.display.set_icon(programIcon)
# ICON=os.path.join(os.getcwd(),"icon.jpg")

# variables
FPS=120
elements=[Element.Element()]
elements[0].isDefault=True
bonds=[Element.Bond()]
bonds[0].type=0
buttons=[Element.Button()]
buttons[0].type=0
relativePos=Element.vec2D(480,290)
selectedElement=elements[0]
selectedBond=bonds[0]
selectedPos=Element.vec2D(0,0)
bufferString=""

# def file_path():
#     filename=
#     return filename

def show_text(text='',x=0,y=0,color=(0,0,0)):
    text=font.render(text,True,color)
```

```
    textRect=text.get_rect()
    textRect.topleft=(x+relativePos.x-10,y+relativePos.y-20)
    screen.blit(text,textRect)

def mouse_click():
    global selectedElement,selectedPos,bufferString
    # add new bond
    operate=False
    t=pygame.mouse.get_pos()
    for element in elements:
        op=element.detect_mouse(Element.vec2D(t[0]-relativePos.x,t[1]-
relativePos.y))
        if op==0:
            element.selected=0
            element.highlight=False
            continue
        elif op==1:
            if selectedElement.id==element.id and not element.left:
                newElement=Element.Element(Element.vec2D(element.pos.x-
100,element.pos.y))
                newBond=Element.Bond(element,newElement)
                # element.left=newElement.right=True
                bonds.append(newBond)
                elements.append(newElement)
            else:
                selectedElement.selected=0
                newBond=Element.Bond(element,selectedElement)
                bonds.append(newBond)
        elif op==2:
            if selectedElement.id==element.id and not element.right:
                newElement=Element.Element(Element.vec2D(element.pos.x+1
00,element.pos.y))
                newBond=Element.Bond(element,newElement)
                # element.right=newElement.left=True
                bonds.append(newBond)
                elements.append(newElement)
            else:
                newBond=Element.Bond(element,selectedElement)
```

```
                bonds.append(newBond)
        elif op==3:
            if selectedElement.id==element.id and not element.up:
                newElement=Element.Element(Element.vec2D(element.pos.x,e
lement.pos.y-100))
                newBond=Element.Bond(element,newElement)
                # element.up=newElement.down=True
                bonds.append(newBond)
                elements.append(newElement)
            else:
                selectedElement.selected=0
                newBond=Element.Bond(element,selectedElement)
                bonds.append(newBond)
        elif op==4:
            if selectedElement.id==element.id and not element.down:
                newElement=Element.Element(Element.vec2D(element.pos.x,e
lement.pos.y+100))
                newBond=Element.Bond(element,newElement)
                # element.down=newElement.up=True
                bonds.append(newBond)
                elements.append(newElement)
            else:
                selectedElement.selected=0
                newBond=Element.Bond(element,selectedElement)
                bonds.append(newBond)
        elif op==5:
            # element is choosed
            if element.highlight:
                element.highlight=False
                selectedPos=Element.vec2D(0,0)
            else:
                element.highlight=True
                bufferString=""
                selectedPos=Element.vec2D(t[0]-relativePos.x,t[1]-
relativePos.y)
        elif op==6 and element.highlight:
            if not element.isDefault:
                for bond in bonds:
```

```
                    if bond.ste.id==element.id or
bond.ede.id==element.id:
                        bond.ste.left=bond.ste.right=bond.ste.up=bond.ste
.down=False
                        bond.ede.left=bond.ede.right=bond.ede.up=bond.ede
.down=False
                        bond.type=0
                    print(bond.ste.id,bond.ede.id,element.id)
                elements.remove(element)
        if op>=1 and op<=6:
            operate=True
        element.selected=0
    selectedElement=Element.Element()
    Element.id-=1
    selectedElement.isDefault=True

    for button in buttons:
        op=button.detect_mouse(Element.vec2D(t[0]-relativePos.x,t[1]-
relativePos.y))
        if op:
            if button.text=="|":
                button.bond.type=1
            elif button.text=="||":
                button.bond.type=2
            elif button.text=="|||":
                button.bond.type=3
            elif button.text=="x":
                button.bond.type=0

    if operate:
        return

    buttons.clear()
    for bond in bonds:
        op=bond.detect_mouse(Element.vec2D(t[0]-relativePos.x,t[1]-
relativePos.y))
        if op:
```

```
            newButton=Element.Button("|",Element.vec2D(t[0]-
relativePos.x,t[1]-relativePos.y+20),[0,0,0],bond)
            buttons.append(newButton)
            newButton=Element.Button("||",Element.vec2D(t[0]-
relativePos.x+20,t[1]-relativePos.y+20),[0,0,0],bond)
            buttons.append(newButton)
            newButton=Element.Button("|||",Element.vec2D(t[0]-
relativePos.x+40,t[1]-relativePos.y+20),[0,0,0],bond)
            buttons.append(newButton)
            newButton=Element.Button("x",Element.vec2D(t[0]-
relativePos.x+60,t[1]-relativePos.y+20),[0,0,0],bond)
            buttons.append(newButton)


    if t[0]<40 and t[0]>0 and t[1]<20 and t[0]>0:
        file=fd.asksaveasfile(filetypes=(('png files', '*.png'),('jpeg
files', '*.jpeg')))
        if file==None:
            file.close()
            return
        file_path=fd.askopenfilename()
        # tp=io.BytesIO()
        f=pygame.image.tostring(screen,"RGBA")
        # tp=list(f)
        img=Image.frombytes("RGBA",(1000,600),f)
        img.save(file_path)

        # image_str = pygame.image.tostring(screen,"RGB")
        # compressed_image_str = zlib.compress(image_str)

        # png.from_array('L').save
        # print(tp)
        # file.write(tp)

def add_bond_only():
    global selectedElement,selectedPos
    if selectedElement.selected!=0:
        return
```

```
    for element in elements:
        t=pygame.mouse.get_pos()
        op=element.detect_mouse(Element.vec2D(t[0]-relativePos.x,t[1]-
relativePos.y))
        if op==0:
            element.selected=0
            continue
        elif op==1 and not element.left:
            element.selected=1
            selectedElement=element
        elif op==2 and not element.right:
            element.selected=2
            selectedElement=element
        elif op==3 and not element.up:
            element.selected=3
            selectedElement=element
        elif op==4 and not element.down:
            element.selected=4
            selectedElement=element



# main loop

InGame=True
while InGame:
    screen.fill((255,255,255))

    # show elements
    for element in elements:
        show_text(element.text,element.pos.x,element.pos.y)
        if element.highlight:
            pygame.draw.rect(screen,(0,0,0),[element.pos.x+relativePos.x
-20,element.pos.y+relativePos.y-20,45,45],1)
        # print(element.id)

    # show button when mouse get close
    t=pygame.mouse.get_pos()
    for element in elements:
```

```
        op=element.detect_mouse(Element.vec2D(t[0]-relativePos.x,t[1]-
relativePos.y))
        if op==0:
            continue
        elif op==1:
            # debug
            pygame.draw.rect(screen,(255,0,0),[element.pos.x+relativePos
.x-40,element.pos.y+relativePos.y-20,20,45],1)
        elif op==2:
            # debug
            pygame.draw.rect(screen,(0,255,0),[element.pos.x+relativePos
.x+25,element.pos.y+relativePos.y-20,20,45],1)
        elif op==3:
            # debug
            pygame.draw.rect(screen,(0,0,255),[element.pos.x+relativePos
.x-20,element.pos.y+relativePos.y-40,45,20],1)
        elif op==4:
            # debug
            pygame.draw.rect(screen,(255,0,255),[element.pos.x+relativeP
os.x-20,element.pos.y+relativePos.y+25,45,20],1)
        elif op==5 and element.highlight and
pygame.mouse.get_pressed()[0]:
            # t=pygame.mouse.get_pos()
            if selectedPos!=Element.vec2D(0,0):
                difference=Element.vec2D((t[0]-relativePos.x)-
selectedPos.x,(t[1]-relativePos.y)-selectedPos.y)
                # print(difference.get_tuple())
                element.pos+=difference
            selectedPos=Element.vec2D(t[0]-relativePos.x,t[1]-
relativePos.y)
        elif op==6 and element.highlight:
            # debug
            pygame.draw.rect(screen,(0,255,255),[element.pos.x+relativeP
os.x+25,element.pos.y+relativePos.y-40,20,20],1)
        # print(op)

    # show button
    for button in buttons:
```

```
        if button.type==0:
            continue
        ft=pygame.font.SysFont('cambriamath',14)
        text=ft.render(button.text,True,button.color)
        textRect=text.get_rect()
        textRect.topleft=(button.pos.x+relativePos.x,button.pos.y+relati
vePos.y)
        pygame.draw.rect(screen,(0,0,0),[button.pos.x+relativePos.x-
5,button.pos.y+relativePos.y,20,20],1)
        screen.blit(text,textRect)

    for bond in bonds:
        if bond.type==0:
            bonds.remove(bond)
    # show bond
    for bond in bonds:
        if bond.type==0:
            continue
        v=Element.vec2D(bond.ede.pos.x-bond.ste.pos.x,bond.ede.pos.y-
bond.ste.pos.y)
        v*=0.3
        if Element.dis(Element.vec2D(0,0),v)>50:
            v.set(v.x,v.y,50)
        n=Element.vec2D(v.y,-v.x)
        n.set(n.x,n.y,5)

        st=Element.vec2D(bond.ste.pos.x,bond.ste.pos.y)
        st+=relativePos
        st+=v
        ed=Element.vec2D(bond.ede.pos.x,bond.ede.pos.y)
        ed+=relativePos
        ed-=v

        if bond.type>=1:
            pygame.draw.line(screen,(0,0,0),st.get_tuple(),ed.get_tuple(
))
        if bond.type>=2:
            st-=n; ed-=n
```

```
            pygame.draw.line(screen,(0,0,0),st.get_tuple(),ed.get_tuple(
))
        if bond.type==3:
            st+=n*2; ed+=n*2
            pygame.draw.line(screen,(0,0,0),st.get_tuple(),ed.get_tuple(
))
    ft=pygame.font.SysFont('cambriamath',14)
    text=ft.render("save",True,(100,100,100))
    textRect=text.get_rect()
    textRect.topleft=(5,0)
    screen.blit(text,textRect)
    pygame.draw.rect(screen,(100,100,100),[0,0,40,20],2)
    pygame.display.flip()

    # event in pygame
    for event in pygame.event.get():
        if event.type==pygame.QUIT:
            InGame=False
        if event.type==pygame.MOUSEBUTTONUP:
            mouse_click()
        if event.type==pygame.MOUSEBUTTONDOWN:
            add_bond_only()
        if event.type==pygame.KEYDOWN:
            keys=pygame.key.get_pressed()
            if keys[pygame.K_BACKSPACE]:
                bufferString=bufferString[:-1]
            else:
                bufferString+=event.unicode
            for element in elements:
                if element.highlight:
                    element.text=bufferString
    clock.tick(FPS)
pygame.quit()
```

Element.py 的程式碼如下。

```
import pygame
import math
import copy
```

```
PI=3.1415926535
EPS=0.00001
id=0
bid=0

class vec2D():
    def __init__(self,dx=0,dy=0):
        self.x = dx
        self.y = dy
    def __deepcopy__(self,memo):
        return vec2D(copy.deepcopy(self._x,memo),
copy.deepcopy(self._y,memo))
    def set(self,dx,dy,length=0):
        l,r = 0.0,1e6
        for i in range(100):
            mid=(l+r)/2
            if (mid*dx)**2+(mid*dy)**2 < length**2:
                l=mid
            else:
                r=mid
        self.x=l*dx
        self.y=l*dy

    def set_angle(self,a,length=0):
        self._x = length*math.cos(a/180*PI)
        self._y = length*math.sin(a/180*PI)

    def __iadd__(self,other):
        self.x+=other.x
        self.y+=other.y
        return self
    def __isub__(self,other):
        self.x-=other.x
        self.y-=other.y
        return self
    def __imul__(self,other):
        self.x*=other
```

```
        self.y*=other
        return self
    def __add__(a,b):
        ret=vec2D(float(a.x),float(a.y))
        ret.x+=b.x
        ret.x+=b.y
        return ret
    def __sub__(a,b):
        ret=vec2D(float(a.x),float(a.y))
        ret.x-=b.x
        ret.x-=b.y
        return ret
    def __mul__(a,b):
        ret=vec2D(float(a.x),float(a.y))
        ret.x*=b
        ret.y*=b
        return ret
    def __eq__(self,other):
        return self.x==other.x and self.y==other.y
    def __ne__(self,other):
        return ~(self==other)
    def get_tuple(self):
        return (self.x,self.y)

def dis(a,b):
    return math.sqrt((a.x-b.x)**2+(a.y-b.y)**2)

def dot(a,b):
    return a.x*b.x+a.y*b.y

class Element:
    def __init__(self,pos=vec2D(0,0)):
        global id
        self.text="C"
        self.pos=pos
        self.left=False
        self.right=False
        self.up=False
```

```
        self.down=False
        self.selected=0
        self.highlight=False
        self.isDefault=False
        self.id=id
        id+=1
    def detect_mouse(self,pos=vec2D(0,0)):
        #
        # 0 2 1
        # left -20,-40,+25,-20
        if pos.x<self.pos.x-20 and pos.x>self.pos.x-40 and
pos.y<self.pos.y+25 and pos.y>self.pos.y-20:
            # left
            return 1
        elif pos.x<self.pos.x+45 and pos.x>self.pos.x+25 and
pos.y<self.pos.y+25 and pos.y>self.pos.y-20:
            # right
            return 2
        elif pos.x<self.pos.x+25 and pos.x>self.pos.x-20 and
pos.y<self.pos.y-20 and pos.y>self.pos.y-40:
            # up
            return 3
        elif pos.x<self.pos.x+25 and pos.x>self.pos.x-20 and
pos.y<self.pos.y+45 and pos.y>self.pos.y+25:
            # down
            return 4
        elif pos.x<self.pos.x+25 and pos.x>self.pos.x-20 and
pos.y<self.pos.y+25 and pos.y>self.pos.y-20:
            # middle
            return 5
        elif pos.x<self.pos.x+45 and pos.x>self.pos.x+25 and
pos.y<self.pos.y-20 and pos.y>self.pos.y-40:
            # right up
            return 6
        return 0

class Bond:
    def __init__(self,ste=Element(vec2D(0,0)),ede=Element(vec2D(0,0))):
```

```
        global bid
        self.ste=ste
        self.ede=ede
        self.type=1
        self.id=bid
        bid+=1
    def detect_mouse(self,pos=vec2D(0,0)):
        # return True or False
        f1=self.ste.pos.y-self.ede.pos.y
        f2=-(self.ste.pos.x-self.ede.pos.x)
        f3=-(self.ste.pos.x*f1+self.ste.pos.y*f2)
        if (pos.x*f1+pos.y*f2+f3)**2<(20**2)*(f1**2+f2**2):
            pb=vec2D(self.ede.pos.x-pos.x,self.ede.pos.y-pos.y)
            pa=vec2D(self.ste.pos.x-pos.x,self.ste.pos.y-pos.y)
            if dot(pa,pb)<=0:
                return True
        return False

class Button:
    def __init__(self,text="click
me",pos=vec2D(0,0),color=[0,0,0],bd=Bond()):
        self.text=text
        self.pos=pos
        self.color=color
        self.type=1
        self.bond=bd
    def detect_mouse(self,pos=vec2D(0,0)):
        if pos.x<self.pos.x+20 and pos.x>self.pos.x and
pos.y<self.pos.y+20 and pos.y>self.pos.y:
            return True
        return False
```

未來展望

1. 使用二元搜尋樹的資料結構，以唯一代碼作為鍵值。如此可以將搜尋元素時的時間複雜度從 $O(n)$降至 $O(\log_2(n))$。
2. 新增繪製芳香烴的功能，芳香烴在有機化合物中具有重要的

地位，有許多的有機化合物有苯環在內。我認為可以建立一個新的物件：苯。裡面有六個 Element 物件，分別有與本體的相對位置座標。苯可以支援旋轉功能，這部分藉由旋轉矩陣實現。而按鍵偵測範圍可以用圓形以節省運算資源。

3. 移動畫布：實現方式與移動元素相似。都是藉由追蹤滑鼠位置與位移來實現。