project_2 (solar,clock,windmill)

20171300 지용환

GIT : https://github.com/YONGYONGA/guaZ/tree/two 의 project2폴더

1. solar_system.

There are four main things to implement in this project. These are the twinkling of stars, the revolution of planets, the revolution of satellites, and spaceships.

First, I'll explain the stars.

```
class stars:
    def __init__(self,cor,color):
        self.center=cor
        self.color=color
        self.cout=1
        self.c=0
    def update(self):
        if(self.cout>100):
            self.c=-10
        elif(self.cout<0):
            self.c=0
            self.cout=0
            self.color=(153,153,0)
        self.cout+=self.c
```

I make star class. In this class, there are variables such as center, color, and cout and c. cout and c are variables for flashing the color for a while and then coming back in the update method.

```
starslist=[]
for i in range(100):
    a=stars([np.random.randint(0,WINDOW_WIDTH),np.random.randint(0,WINDOW_HEIGHT)
    starslist.append(a)
```

This part is just a part of creating a global variable class of 100 stars. The initial color is slightly dark yellow (153,153,0).

In while loop, as soon as initiallize screen black,

```
160     screen.fill(BLACK)
161     for i in range(100):
162         pygame.draw.circle(screen,starslist[i].color,starslist[i].center,2)
163         starslist[i].update()
164     # 안티얼리어스를 적용하고 검은색 문자열 렌더링
165     if for_timer>=100:
166         twinkle()
167         for_timer=1
```

draw 100 stars in screen. and update this. At first time, since c is 0, nothing changes.

But when for_timer becmes 100(this variable is global variable initial value of 1.)

use twinkle method.

```
def twinkle():
    randomlist=[]
    i=1
    while i<=20:
        random=np.random.randint(0,100)
        if random in randomlist:
            continue
        else:
            randomlist.append(random)
            i+=1
    for i in range(20):
        starslist[randomlist[i]].color=YELLOW
        starslist[randomlist[i]].cout=101
```

First, 20 non-overlapping numbers are randomly selected, and the selected numbers are used as indices to change the color of the star at that index to yellow and cout to 101. Then since cout is 101, c becmoes -10 at update, then until cout gets smaller then 0, this star color is yellow. That is, the color of the star briefly turns yellow and then returns. This is how stars twinkle.
The next implementation is the orbit of planets and moons.

```
117   ##각자의 위치,각도(원점이 태양이라 생각
118   #금성
119   ve = np.array([100,0, 1])
120   ve_dg = 10
121   #지구와 달
122   ea=np.array([200,0,1])
123   ea_dg=20
124   mo_dg=20
125   #지구~달의 거리
126   mo=np.array([70,0,1])
127
128   ##토성과 달 타원임이건.
129   xRadius = 450
130   yRadius = 300
131   sa=np.array([300,0,1])
132   sa_dg=20
133   ta_dg=20
134   #공전주기 29년
135   for_cal=360/(29*12*30)
136   #토성~달의 거리
137   ta=np.array([80,0,1])
138
```

Define all plnet and moons location and degree in global variable. In addition, since the orbit of Saturn will be implemented as an ellipse, the radius of the

ellipse is declared.

```
22  def Rmat(degree):
23      radian = np.deg2rad(degree)
24      c = np.cos(radian)
25      s = np.sin(radian)
26      R = np.array( [[ c, -s, 0], [s, c, 0], [0, 0, 1] ] )
27      return R
28
29  def Tmat(a,b):
30      H = np.eye(3)
31      H[0,2] = a
32      H[1,2] = b
33      return H
```

Since this method is used so often, I will only explain it briefly. Returns a rotation transformation matrix and a translation matrix, respectively.
In while loop accurately declares the change in angle for an accurate orbital period. I,e venus's orbital period is 225 days. So, add 360/225 per clock tick.

```
169     screen.blit(im, [WINDOW_WIDTH/2-60, WINDOW_HEIGHT/2-60])
```

This is sun. I blit this image center of the screen. Since this image size is 120*120 must subtart 60,60

```
173     H = Tmat(WINDOW_WIDTH/2,WINDOW_HEIGHT/2) @ Rmat(ve_dg)
174     corp = H @ ve
175     screen.blit(venus, corp[:2]-[20,20])
```

Venus rotates around the sun. So, Just multimle Tranform matrix to center and rotation matrix. And blit this image. Then venus rotate aroud sun.

```
177     ##지구와 달(달은 지구 중심이다.)
178     #pygame.draw.circle(screen,WHITE,[WINDOW_WIDTH/2,WINDOW_HEIGHT/2],2
179     H = Tmat(WINDOW_WIDTH/2,WINDOW_HEIGHT/2) @ Rmat(ea_dg)
180     corp = H @ ea
181     screen.blit(earth, corp[:2]-[35,35])
182     H=Tmat(corp[0],corp[1]) @ Rmat(mo_dg)
183     corp = H @ mo
184     screen.blit(moon, corp[:2]-[10,10])
```

Simialry earth rotate the sun. But moon rotate around Earth. So transformation matrix must contain the position of the center of the earth. This is copr[0] and corp[1]. Then blit this and then moon rotate around Earth.

```
    ##토성과 달
    #타원계산, 그후 달 변환
    x1 = int(math.cos(sa_dg * 2*  math.pi / 360) * xRadius) + WINDOW_WIDTH/2
    y1 = int(math.sin(sa_dg * 2* math.pi / 360) * yRadius) + WINDOW_HEIGHT/2
    #pygame.draw.ellipse(screen, WHITE, [500, 200, 900, 600], 1)
    screen.blit(saturn,[x1-44, y1-44])
    H=Tmat(x1,y1) @ Rmat(ta_dg)
    corp = H @ ta
    screen.blit(taitan, corp[:2]-[10,10])
```

Finally Saturn's orbital path is elliptical. Since it is not possible to use a general rotation transformation matrix, a separate calculation was made. It's simple!

`sa_dg * 2*  math.pi / 360` This part is for conversion of radians and degrees. After transformation, take cos for the x coordinate and sin for the y coordinate and, of course, multiply by the radius of x and the radius of y, respectively. Because it is an ellipse, each is multiplied by a different value. And just like the role of the movement matrix, if you move it to the center by just adding and draw it on that part, it is an elliptical rotation. The moon can be drawn in the same way as the earth's moon. Now we have realized the orbit of 3 planets and 2 satellites. All that remains is the movement of the ship.

```
#우주선 움직이기
dx=2
dy=2
x=220
y=120
```

This is global variable, dx,dy is velocity and x,y is location of starship. In while loop

```
x+=dx
y+=dy
check()
```

add velocity and use check method.

```
def check():
    global dx
    global dy
    if(x<40 or x>WINDOW_WIDTH-20):
        dx*=-1
    elif(y<40 or y>WINDOW_HEIGHT-20):
        dy*=-1
    if(dx<0 and dy<0):
        pic=pygame.transform.rotate(starship,90)
    elif(dx<0 and dy>0):
        pic=pygame.transform.rotate(starship,180)
    elif(dx>0 and dy<0):
        pic=starship
    else:
        pic=pygame.transform.rotate(starship,270)
    screen.blit(pic,[x-20, y-20])
```

This is check method. First if and elif statement implements a collision with a wall. When colliding, it moves in the opposite direction. And second if and elif statemet is for starship rotate. Since the spaceship has a head, the picture was rotated according to the direction of movement to realize further movement realistically.

2. clock

The description of Rmat and Tmat functions will be omitted.

```
spoly = np.array( [[0, 0, 1], [270, 0, 1],[290,10,1], [270, 20, 1], [0, 20, 1]])
spoly = spoly.T # 3x4 matrix
mpoly = np.array( [[0, 0, 1], [200, 0, 1],[220,10,1], [200, 20, 1], [0, 20, 1]])
mpoly = mpoly.T # 3x4 matrix
hpoly = np.array( [[0, 0, 1], [150, 0, 1],[170,10,1], [150, 20, 1], [0, 20, 1]])
hpoly = hpoly.T # 3x4 matrix

cor = np.array([10, 10, 1])
s=datetime.datetime.now()
sec=s.second
min=s.minute
hour=s.hour
degree = 10
degree2=10
degree3=10
screen.fill(WHITE)
drawclock()
```

It creates the shape of the hour hand, minute hand, and second hand as global variables, and receives the hour, minute, and second through datetime.now, and also creates the background of the clock through drawclock.

```
def drawclock():
    pygame.draw.circle(screen,BLACK,[285,285],290,1)
    for i in range(1,13):
        k=i
        text = font.render(str(k), True, BLACK)
        radian=np.deg2rad(30*(k-3))
        if(i==9):
            screen.blit(text, [(np.cos(radian))*290+299, (np.sin(radian))*290+285
        elif(i==12):
            screen.blit(text, [(np.cos(radian))*290+285, (np.sin(radian))*290+299
        else:
            screen.blit(text, [(np.cos(radian))*290+285, (np.sin(radian))*290+285
```

This is drawclock. Draw center and blit number using cos and sin. Since the angle of the clock number is 30 degrees each, I multiplied it by 30 and took k-3 because it is 0 degrees from 3 o'clock due to the nature of pygame.
In while loop

```
82      s=datetime.datetime.now()
83      if(sec!=s.second): ##1초마다 이벤트 발생(초침변경)
```

check current seconds using datetime.now.second and if it's different from the seconds we got in the previous while loop, i.e. the seconds are different, then we just draw everything. In other words, draw everything every second.

```
85          screen.fill(WHITE)
86          drawclock()
87          min=s.minute
88          hour=s.hour
```

Clear screen and drawclock and get minute and hour.

```
93          degree =(sec*6)-90
94          degree2=(min*6)-90
95          degree3=((hour%12)*30)-90
```

Seconds and minutes are 60, i.e., each 6 degrees is multiplied by 6, and an hour is 30 degrees, multiplied by 30. As I said, pygame starts at 90 degrees, so subtract 90.

```
97          H = Tmat(285, 285) @ Rmat(degree)
98          pp = H @ spoly
99          corp = H @ cor
100         # print(pp.shape, pp, pp.T )
101
102         q = pp[0:2, :].T # N x 2 matrix
103         pygame.draw.polygon(screen, RED, q, 4)
104         pygame.draw.circle(screen, (255, 128, 128), corp[:2], 3)
105
```

This is secnod hand. Using Tmat and Rmat draw second hand. Since we have

already saved the angle by figuring out the time, just get it and draw it and you're done.

```
106         #분침
107         H = Tmat(285, 285) @ Rmat(degree2)@Tmat(0,-5)
108         pp = H @ mpoly
109
110         q = pp[0:2, :].T # N x 2 matrix
111         pygame.draw.polygon(screen, BLACK, q, 4)
112         #시침
113         H = Tmat(285, 285) @ Rmat(degree3)@Tmat(-8,0)
114         pp = H @ hpoly
115         q = pp[0:2, :].T # N x 2 matrix
```

If the minute and hour hands are drawn in the same way, a clock is realized.
Final project is windmil.

3. Windmill

```
53  connect=np.array( [[0, 0, 1], [40, 0, 1], [40, 10, 1], [0, 10, 1]])
54  connect=connect.T
55  joint=np.array([33,5,1])
56  # 날개
57  poly = np.array( [[0, 0, 1], [200, 0, 1], [200, 59, 1], [0, 59, 1]])
58  poly = poly.T # 3x4 matrix
59
60  cor = np.array([10, 10, 1])
61
62  cor1 = np.array([70, 0, 1])
63  cor11 = np.array([70, 59, 1])
64  cor2 = np.array([140, 0, 1])
65  cor22 = np.array([140, 59, 1])
66  cor3= np.array([0,30 , 1])
67  cor33= np.array([200, 30, 1])
```

This is just shape. The connect is connected to the windmill and windmill blades are attached to connect. All cor1, cor11... vairables is line that was declared to make the wings look a bit prettier. In while,loop

```
keystate = pygame.key.get_pressed()
if keystate[pygame.K_UP]:
    speed+=0.5
elif keystate[pygame.K_DOWN]:
    speed-=0.5
wind_dg+=speed
```

If we press up key then speed goes up, and vice versa, the speed goes down. I draw 4 winds and each wing is 90 degrees apart, so I only use one angle and add 90 degrees to it.

```
121     H = Tmat(300, 350) @ Rmat(wind_dg)
122     pp = H @ connect
123     joint11=H@joint
124     q = pp[0:2, :].T # N x 2 matrix
125     pygame.draw.polygon(screen, RED, q)
126     pygame.draw.circle(screen, (255, 128, 128), joint11[:2], 3)
127
128     H = Tmat(300, 350) @ Rmat(wind_dg+90)
129     pp = H @ connect
130     joint22=H@joint
131     q = pp[0:2, :].T # N x 2 matrix
132     pygame.draw.polygon(screen, RED, q)
133     pygame.draw.circle(screen, (255, 128, 128), joint22[:2], 3)
134
```

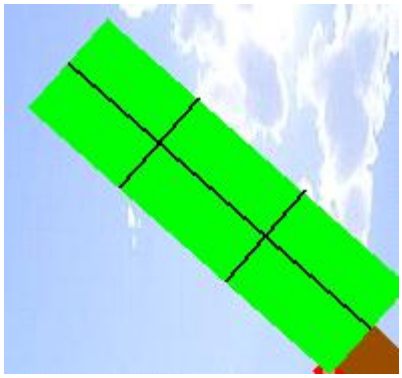
Since center is 300,350 using Tmat, move 300,350 and rotate using Rmat. Everything else is the same, but since the angular plane is 90 degrees different, you can implement it with wing_dg + 90 as shown below. Next connect connect and wings.

```
150     H = Tmat(joint11[0],joint11[1]) @ Rmat(wind_dg)
151     pp = H @ poly
152     q = pp[0:2, :].T # N x 2 matrix
153     pygame.draw.polygon(screen, GREEN, q)
154     muls(H)
155
156     H = Tmat(joint22[0],joint22[1]) @ Rmat(wind_dg+90)
157     pp = H @ poly
158     q = pp[0:2, :].T # N x 2 matrix
159     pygame.draw.polygon(screen, GREEN, q)
160     muls(H)
```

Each connect has joint11 using this Tmat makes the movement to the position of the joint. And, draw wings. The wing and connect then rotate together while remaining connected. Using muls, darw just line in wing.

```
85  def muls(k):
86      one=k@cor1
87      two=k@cor11
88      three=k@cor2
89      four=k@cor22
90      five=k@cor3
91      six=k@cor33
92      pygame.draw.line(screen,BLACK,one[:2],two[:2],2)
93      pygame.draw.line(screen,BLACK,three[:2],four[:2],2)
94      pygame.draw.line(screen,BLACK,five[:2],six[:2],2)
```

This is muls. just multiple matrix with cor, and draw this line



Then we can draw 3 lines like this.

```
text = font.render("NOW SEEPD : {}".format(speed), True, BLACK)
screen.blit(text, [100, 30])
text = font1.render("Adjust the speed using the up and down keys", True, BLACK
screen.blit(text, [20, 70])
```

Finally just blit speed information text. Then windnmill is over.