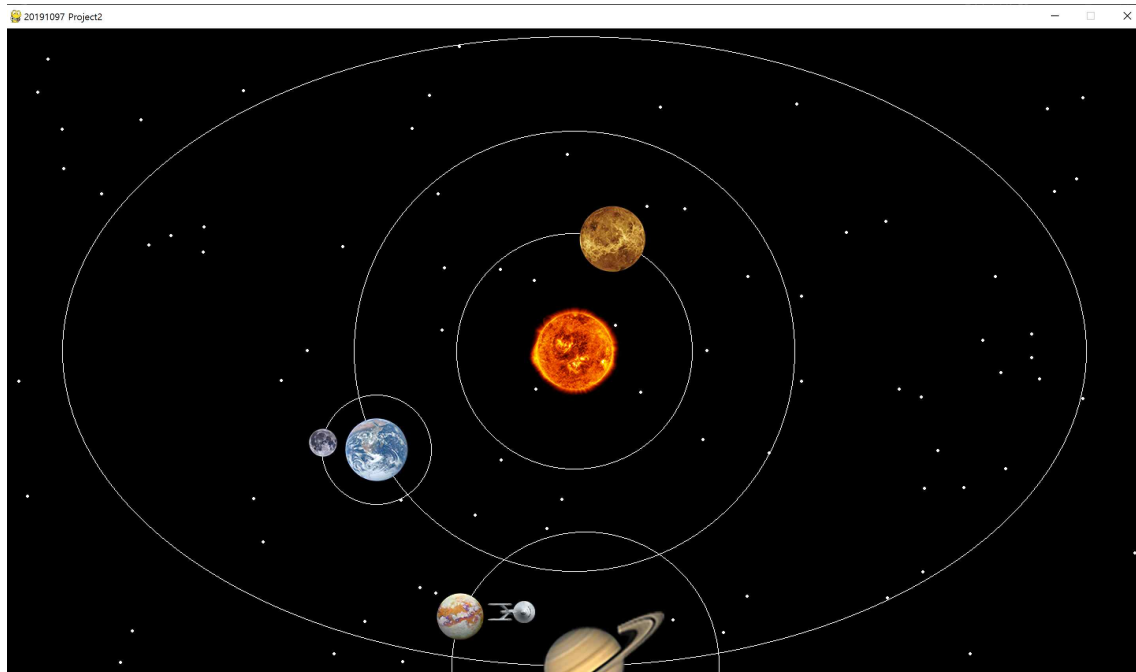


Explanation of Project 2

20191097 배호용

github - <https://github.com/SGU-20191097-BHY/20191097-Project2>

1.Solar system



This is program running. Planets rotate around the Sun counter-clockwise. 80 stars randomly appear and randomly appear again every 800 ticks. Enterprise wanders the window. It changes its direction and speed every 1800 ticks.

```
31 class Planet():
32     def __init__(self, centerxy, distance, cycle, image):
33         self.centercoord = centerxy
34         self.dist=distance
35         self.rot=0
36         self.rotspeed = 360 / cycle / 40
37         self.image = image
38         self.centerx=centerxy[0]
39         self.centery=centerxy[1]
40         self.x=self.centerx + self.dist
41         self.y=self.centery
42
43     def update(self, centerplanet):
44         self.centerx=centerplanet.x
45         self.centery=centerplanet.y
46         self.x=self.dist * np.sin(self.rot) + self.centerx
47         self.y=self.dist * np.cos(self.rot) + self.centery
48
49         self.rot += self.rotspeed
50
51     def show(self,):
52         self.width=self.image.get_width()
53         self.height=self.image.get_height()
54         pos=[self.x-self.width/2, self.y-self.height/2]
55         pygame.draw.circle(screen,WHITE, [self.centerx,self.centery],self.dist, 1)
56         screen.blit(self.image, pos)
57
```

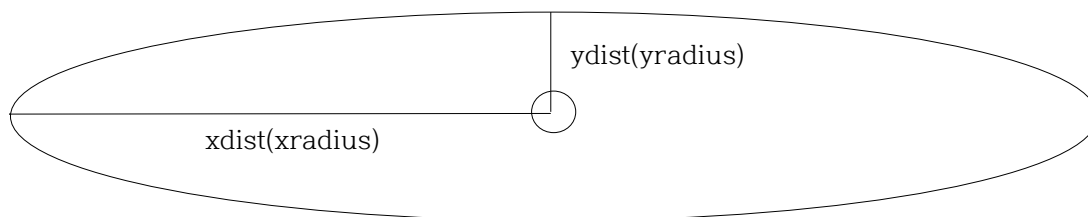
I didn't used matrix. Just rotating a dot and showing a image at that dot is much simpler.

Before line 31, I imported modules, set window, loaded image files, defined colors. I made Planet class for circular orbit planets. Planet object has it's rotation speed, image file, xy of orbit's center, radius(distance) of orbit.

Update method updates planet's x and y according to its cycle. Show method blits image on the screen. xy of the planet becomes center of the image.

```
58 class Saturn():
59     def __init__(self,centerxy, xdist, ydist, cycle, image):
60         self.centercoord=centerxy
61         self.centerx=centerxy[0]
62         self.centery=centerxy[1]
63         self.xradius=xdist
64         self.yradius=ydist
65         self.rot=0
66         self.rotspeed = 360 / cycle / 40
67         self.image=image
68         self.x=xdist+self.centerx
69         self.y=ydist+self.centery
70
71     def update(self, centerplanet):
72         self.centerx=centerplanet.x
73         self.centery=centerplanet.y
74         self.x=self.xradius * np.sin(self.rot) + self.centerx
75         self.y=self.yradius * np.cos(self.rot) + self.centery
76
77         self.rot += self.rotspeed
78
79     def show(self,):
80         self.width=self.image.get_width()
81         self.height=self.image.get_height()
82         pos=[self.x-self.width/2, self.y-self.height/2]
83         pygame.draw.ellipse(screen,WHITE, [self.centerx-self.xradius,self.centery-self.yradius,
84             self.xradius*2,self.yradius*2], 1)
85         screen.blit(self.image, pos)
```

And I made saturn class for Saturn's elliptical orbit. Other things are same as planet class. xdist and ydist means below.



Update mothod updates saturn's xy. I had no idea how to calculate elliptical movement, so I googled it...

Show method is same as planet's. Only draw.ellipse() is different.

```

87 class starship():
88     def __init__(self, image):
89         self.image=image
90         self.width=self.image.get_width()
91         self.height=self.image.get_height()
92         self.x=np.random.randint(1+self.width, WINDOW_WIDTH-self.width)
93         self.y=np.random.randint(1+self.height, WINDOW_HEIGHT-self.height)
94         self.last_update = 0
95         self.dx=np.random.randint(-5,5)
96         self.dy=np.random.randint(-5,5)
97
98     def update(self):
99         if pygame.time.get_ticks() > self.last_update +1800:
100             self.dx=np.random.randint(-5,5)
101             self.dy=np.random.randint(-5,5)
102             self.last_update = pygame.time.get_ticks()
103         self.x += self.dx
104         self.y += self.dy
105         if self.x > WINDOW_WIDTH or self.x < 0:
106             self.dx *= -1
107         if self.y > WINDOW_HEIGHT or self.y < 0:
108             self.dy *= -1
109
110     def show(self,):
111         pos=[self.x-self.width/2, self.y-self.height/2]
112         screen.blit(self.image, pos)
113

```

This is class for starship Enterprise. Starship's first xy is randomly determined, and the image doesn't get out of the window. And it's velocity is determined randomly.

Update method updates starship's velocity every 1800 ticks. And if the starship pass the window border, it bounces back to window.

Show method blits Enterprise image on screen.

```

114 sun=Planet([WINDOW_WIDTH/2, WINDOW_HEIGHT/2],0,1,sun_image)
115 earth=Planet([sun.x, sun.y], 280, 365, earth_image)
116 venus=Planet([sun.x, sun.y], 150, 225, venus_image)
117 moon=Planet([earth.x, earth.y],70,27,moon_image)
118 saturn=Saturn([sun.x,sun.y],650,400,10749,saturn_image)
119 titan=Planet([saturn.x,saturn.y],170,15.9,titan_image)
120
121 planets=[sun,earth,venus,saturn]
122 satellites=[moon,titan]
123 enterprise=starship(enterprise_image)
124

```

And I made objects. Orbital period is how many days take to 1 cycle. Sun is fixed at center of window. Earth, Venus, Saturn rotates around the Sun. And Moon rotates around Earth, and Titan rotates around Saturn.

```

125 last_update=-800
126 |
127 done=False
128 while not done:
129     for event in pygame.event.get():
130         if event.type == pygame.QUIT:
131             done=True
132
133     screen.fill(BLACK)
134     if last_update + 800 < pygame.time.get_ticks():
135         stars=[]
136         for i in range(80):
137             stars.append([np.random.randint(0,WINDOW_WIDTH), np.random.randint(0,WINDOW_HEIGHT)])
138             last_update=pygame.time.get_ticks()
139
140     for i in stars:
141         pygame.draw.circle(screen,WHITE, i, 2, 0)
142

```

This is code for background stars. Every 800 ticks, 80 random xys generated and little white circle is drawn at that xy. These circles look like stars.

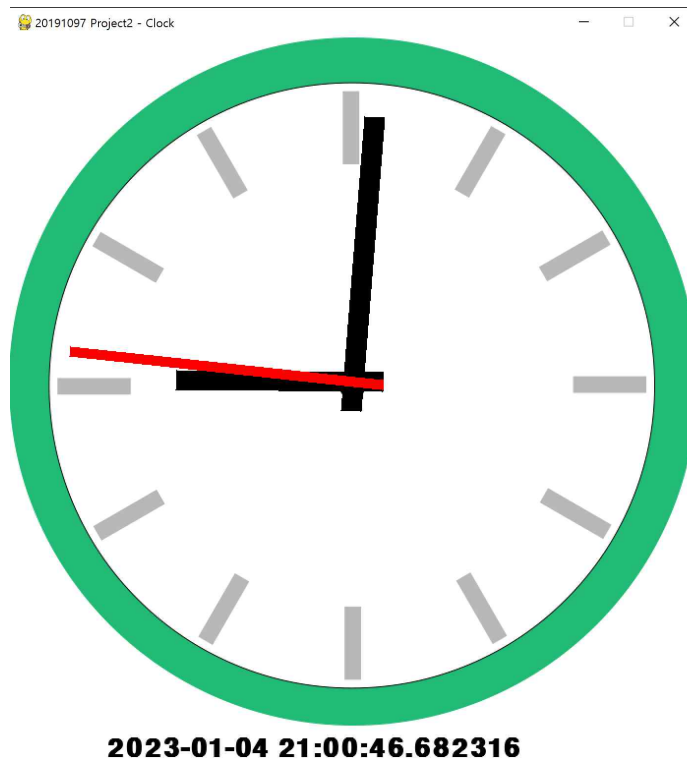
```

143 v for i in planets:
144     i.update(sun)
145     i.show()
146
147     moon.update(earth)
148     titan.update(saturn)
149     moon.show()
150     titan.show()
151
152     enterprise.update()
153     enterprise.show()
154
155     pygame.display.flip()
156     clock.tick(120)
157

```

And I updated each objects and used show method to blit on screen.

2.Clock



This is my program. Like a real clock, hour hand and minute hand moves slightly at every second. Below the clock, real time is being printed and updated.

```
37 def Rmat(deg):
38     radian = np.deg2rad(deg)
39     c=np.cos(radian)
40     s=np.sin(radian)
41     R=np.array([[c, -s, 0], [s,c,0], [0,0,1]])
42     return R
43
44 def Tmat(a,b):
45     H=np.eye(3)
46     H[0,2]=a
47     H[1,2]=b
48     return H
49
50 hourpoly=np.array([ [0,0,1],[210,0,1],[210,20,1],[0,20,1] ])
51 hourpoly1=hourpoly.T
52 minutepoly=np.array([ [0,0,1],[300,0,1],[300,20,1],[0,20,1] ])
53 minutepoly1=minutepoly.T
54 secondpoly=np.array([ [0,0,1],[320,0,1],[320,10,1],[0,10,1] ])
55 secondpoly1=secondpoly.T
56
57 cor1=np.array([30,10,1])
58 cor2=np.array([30,5,1])
59
```

Before line 37, I imported modules, defined assets and its paths, defined colors and window.

Rmat returns rotated matrix. Tmat translates matrix.

Hourpoly is hour hand, minute poly is minute hand and second poly is second hand. Hour poly is shortest, and second poly is longest and thinnest. cor1 is center of rotation for hour hand and minute hand. cor2 is for second hand because it's thinner than the others.

```
64 done=False
65 while not done:
66     screen.fill(WHITE)
67     screen.blit(clockbackground,[0,0])
68     # 이벤트 반복 구간
69     for event in pygame.event.get():
70         if event.type == pygame.QUIT:
71             done = True
72     timenow=datetime.datetime.now()
73     degree1 = timenow.hour*30 -90 + timenow.minute/2 + timenow.second/120
74     degree2 = timenow.minute*6 + timenow.second/10 -90
75     degree3 = timenow.second*6 -90
76
77     hourH=Tmat(320,340) @ Tmat(30,10) @ Rmat(degree1) @ Tmat(-30,-10)
78     hourpp = hourH @ hourpoly1
79     hourcor = hourH @ cor1
80     hour=hourpp[0:2,:].T
81
82     cor= hourH @ cor1
83     minuteH=Tmat(320,340) @ Tmat(30,10) @ Rmat(degree2) @ Tmat(-30,-10)
84     minutepp = minuteH @ minutepoly1
85     minute=minutepp[0:2,:].T
86
87     secondH=Tmat(320,345) @ Tmat(30,5) @ Rmat(degree3) @ Tmat(-30,-5)
88     secondcor= secondH @ cor2
89     secondpp = secondH @ secondpoly1
90     second=secondpp[0:2,:].T
91
92     pygame.draw.polygon(screen, BLACK, hour, 0)
93     pygame.draw.polygon(screen, BLACK, minute, 0)
94     pygame.draw.polygon(screen, RED, second, 0)
95     time=font.render(str(datetime.datetime.now()),True,BLACK)
96     screen.blit(time,[100,710])
97
```

Line 72 ~ 75 calculates the degrees for each hands. They moves slightly in every second.

*1 hour (30 degrees) = 60 minute (60*6 degrees) = 3600 second(60*60*6 degrees)*

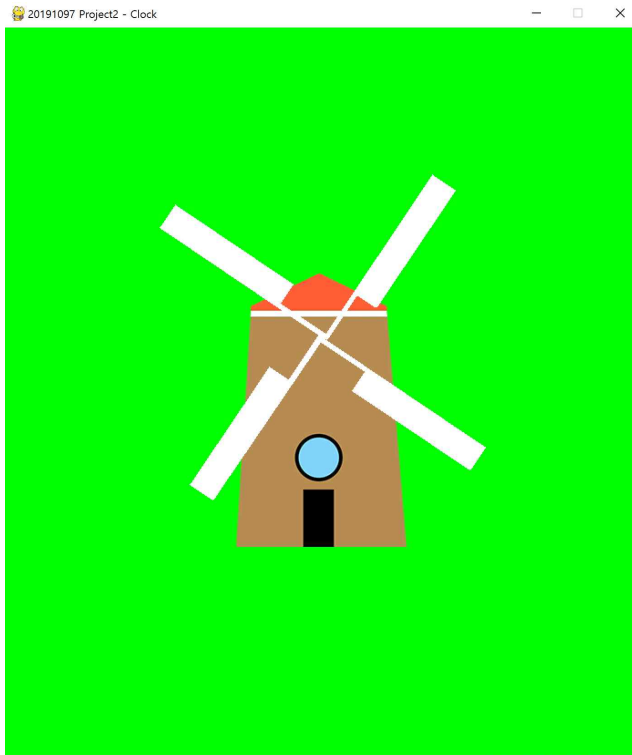
*1/60hour (30/60 degrees) = 1 minute(6 degrees) = 60 second(60*6 degrees)*

1/3600hour (1/120 degrees) = 1/60 minute (6/60 degrees) = 1 second(6 degrees)

So, at every second, hourhand moves 1/120 degrees, minute and moves 1/10 degrees, second hand moves 6 degrees.

Line 77 ~ 90 translates and rotates hand-polys. Line 92 ~ 96 visualizes them.

3.Windmill



This is my windmill. Wings rotate counter-clockwise.

```
48 wingpoly=np.array([ [0,0,1],[215,0,1],[215,30,1],[60,30,1],[60,5,1],[0,5,1] ])
49 wing=wingpoly.T
50 degree = 0
```

Before these lines is same as clock file. In these lines I made polygon of a wing. It looks like below. It is OK to leave the center of rotation (0,0).

(0,0)



```
52 done=False
53 while not done:
54     screen.fill(GREEN)
55     screen.blit(clockbackground,[255,270])
56     for event in pygame.event.get():
57         if event.type == pygame.QUIT:
58             done = True
59
60     degree-=1
61
62     wing1H=Tmat(350,340) @ Rmat(degree)
63     wing1pp = wing1H @ wing
64     wing1=wing1pp[0:2,:].T
65
66     wing2H=Tmat(350,340) @ Rmat(degree+90)
67     wing2pp = wing2H @ wing
68     wing2=wing2pp[0:2,:].T
69
70     wing3H=Tmat(350,340) @ Rmat(degree+180)
71     wing3pp = wing3H @ wing
72     wing3=wing3pp[0:2,:].T
73
74     wing4H=Tmat(350,340) @ Rmat(degree+270)
75     wing4pp = wing4H @ wing
76     wing4=wing4pp[0:2,:].T
77
```

So, I only used one Tmat to move wing to windmill building, and Rmat to rotate the wing. Each wing has 90 degrees difference.