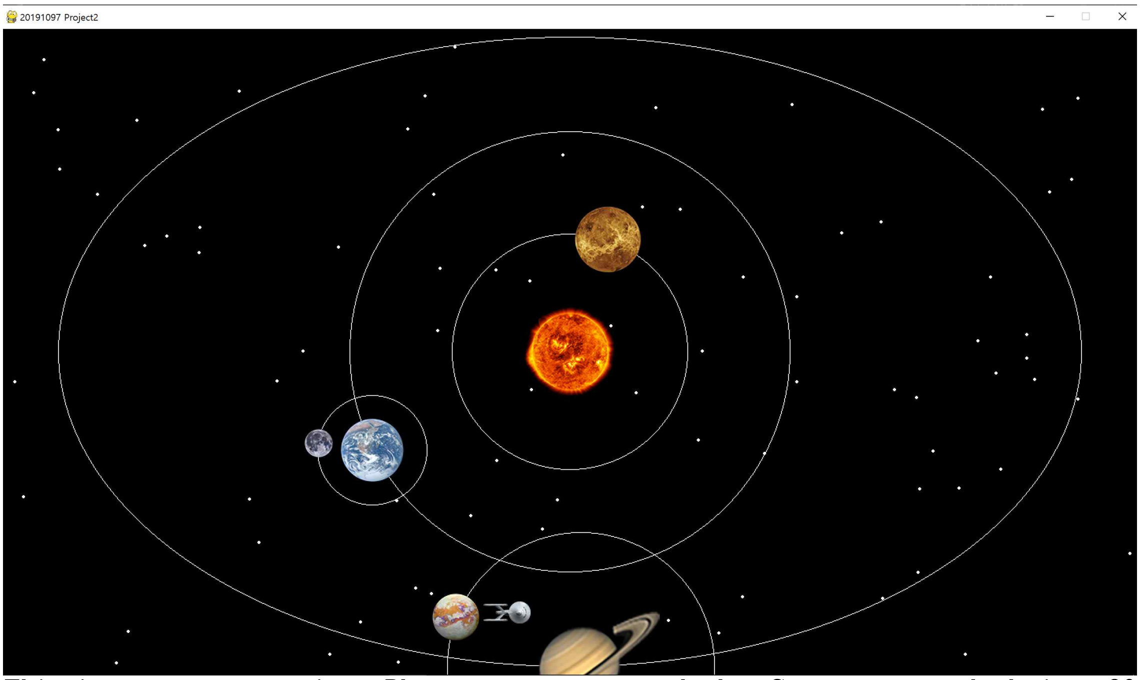
Explanation of Project 2

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.

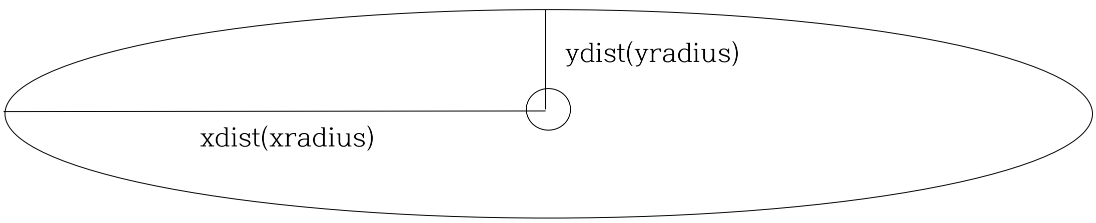
Before line 31, I imported modules, set window, loaded image files, defined colors.

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.

```
class Saturn():
    def __init__(self,centerxy, xdist, ydist, cycle, image):
        self.centercoord=centerxy
        self.centerx=centerxy[0]
        self.centery=centerxy[1]
        self.xradius=xdist
        self.yradius=ydist
        self.rot=0
        self.rotspeed = 360 / cycle / 40
        self.image=image
        self.x=xdist+self.centerx
        self.y=ydist+self.centery

    def update(self, centerplanet):
        self.centerx=centerplanet.x
        self.centery=centerplanet.y
        self.x=self.xradius * np.sin(self.rot*2*np.pi/360) + self.centerx
        self.y=self.yradius * np.cos(self.rot*2*np.pi/360) + self.centery

        self.rot += self.rotspeed

    def show(self,):
        self.width=self.image.get_width()
        self.height=self.image.get_height()
        pos=[self.x-self.width/2, self.y-self.height/2]
        pygame.draw.ellipse(screen,WHITE, [self.centerx-self.xradius,self.centery-self.yradius,
            self.xradius*2,self.yradius*2], 1)
        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 to calculate elliptical movement, so I googled it...

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

```python
class starship():
    def __init__(self, image):
        self.image=image
        self.width=self.image.get_width()
        self.height=self.image.get_height()
        self.x=np.random.randint(1+self.width, WINDOW_WIDTH-self.width)
        self.y=np.random.randint(1+self.height, WINDOW_HEIGHT-self.height)
        self.last_update = 0
        self.dx=np.random.randint(-5,5)
        self.dy=np.random.randint(-5,5)

    def update(self):
        if pygame.time.get_ticks() > self.last_update +1800:
            self.dx=np.random.randint(-5,5)
            self.dy=np.random.randint(-5,5)
            self.last_update = pygame.time.get_ticks()
        self.x += self.dx
        self.y += self.dy
        if self.x > WINDOW_WIDTH or self.x < 0:
            self.dx *= -1
        if self. y > WINDOW_HEIGHT or self.y < 0:
            self.dy *= -1

    def show(self,):
            pos=[self.x-self.width/2, self.y-self.height/2]
            screen.blit(self.image, pos)
```

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.

```python
sun=Planet([WINDOW_WIDTH/2, WINDOW_HEIGHT/2],0,1,sun_image)
earth=Planet([sun.x, sun.y], 280, 365, earth_image)
venus=Planet([sun.x, sun.y], 150, 225, venus_image)
moon=Planet([earth.x, earth.y],70,27,moon_image)
saturn=Saturn([sun.x,sun.y],650,400,10749,saturn_image)
titan=Planet([saturn.x,saturn.y],170,15.9,titan_image)

planets=[sun,earth,venus,saturn]
satelites=[moon,titan]
enterprise=starship(enterprise_image)
```

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.

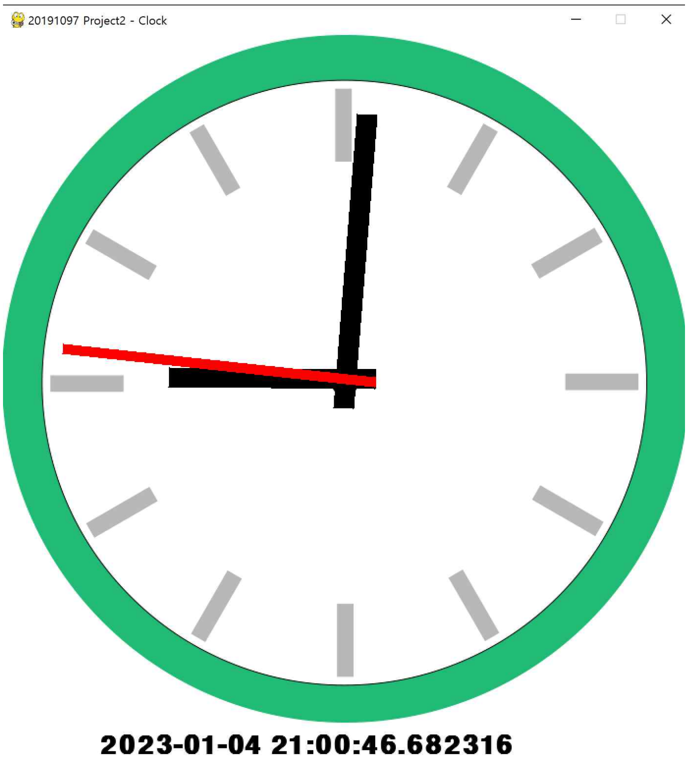
```python
last_update=-800

done=False
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done=True

    screen.fill(BLACK)
    if last_update + 800 < pygame.time.get_ticks():
        stars=[]
        for i in range(80):
            stars.append([np.random.randint(0,WINDOW_WIDTH), np.random.randint(0,WINDOW_HEIGHT)])
            last_update=pygame.time.get_ticks()

    for i in stars:
        pygame.draw.circle(screen,WHITE, i, 2, 0)
```

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.

```python
    for i in planets:
        i.update(sun)
        i.show()

    moon.update(earth)
    titan.update(saturn)
    moon.show()
    titan.show()

    enterprise.update()
    enterprise.show()

    pygame.display.flip()
    clock.tick(120)

```

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])
```

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 caculates 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)*

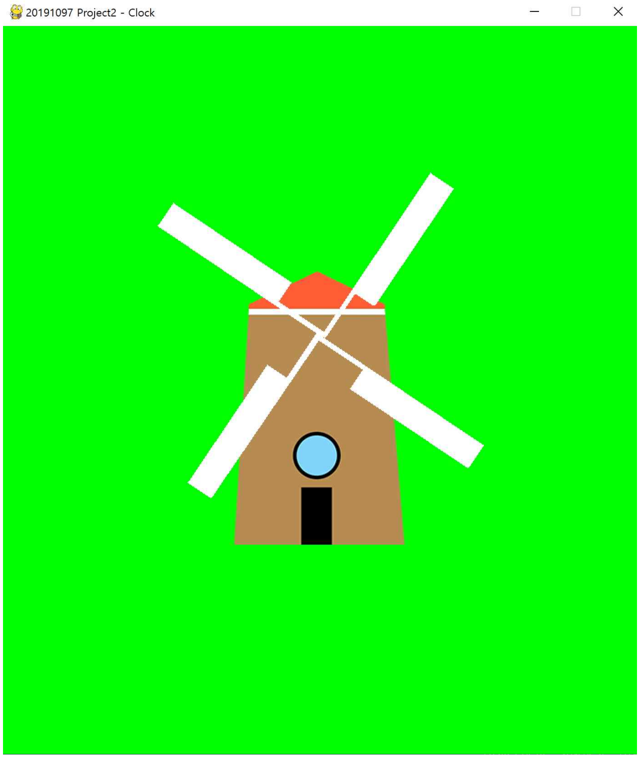
*1/60hour (30/60 degrees) = 1 minute(6 degrees) = 60 second(60*6 degrees)*

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.

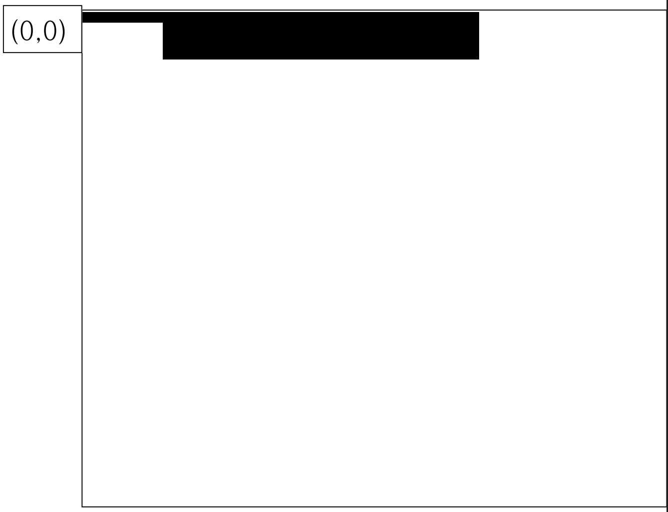
3.Windmill



This is my windmill. Wings rotate counter-clockwise.

```python
wingpoly=np.array([ [0,0,1],[215,0,1],[215,30,1],[60,30,1],[60,5,1],[0,5,1] ])
wing=wingpoly.T
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).




```python
done=False
while not done:
    screen.fill(GREEN)
    screen.blit(clockbackground,[255,270])
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    degree-=1

    wing1H=Tmat(350,340) @ Rmat(degree)
    wing1pp = wing1H @ wing
    wing1=wing1pp[0:2,:].T

    wing2H=Tmat(350,340) @ Rmat(degree+90)
    wing2pp = wing2H @ wing
    wing2=wing2pp[0:2,:].T

    wing3H=Tmat(350,340) @ Rmat(degree+180)
    wing3pp = wing3H @ wing
    wing3=wing3pp[0:2,:].T

    wing4H=Tmat(350,340) @ Rmat(degree+270)
    wing4pp = wing4H @ wing
    wing4=wing4pp[0:2,:].T
```

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