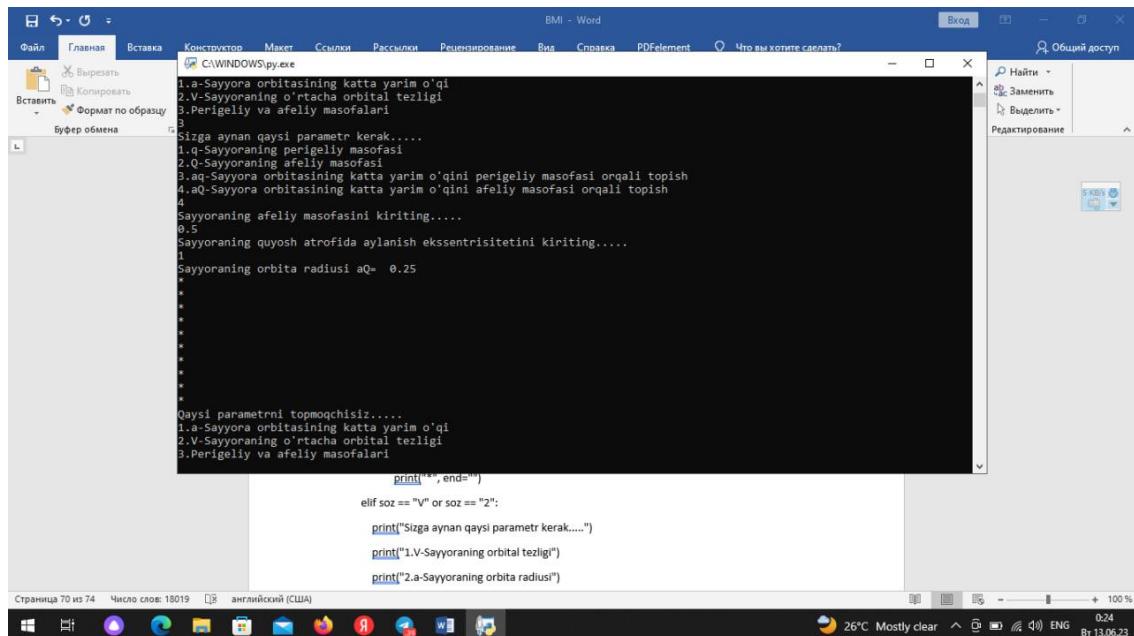


ISSN (E): 2181-4570

PYTON DASTURIDA ASTRONOMIYADAN ANIMATSIYA YARATISH

Haydarova Dilorom- Navoiy davlat pedagogika instituti magistranti
Ilmiy rahbar: Sayfullayeva Gulhayo- Navoiy davlat pedagogika instituti dotsenti



```

Файл Главная Вставка Конструктор Макет Ссылки Рассылки Рецензирование Вид Справка PDFelement Что вы хотите сделать? Вход Общий доступ
Создать документ Вырезать Копировать Формат по образцу Буфер обмена
1.a-Sayyora orbitasining katta yarim o'qi
2.V-Sayyoranor o'ttacha orbital tezligi
3.Perigeliy va afeliy masofalari
3
Sizga aynan qaysi parametr kerak.....
1.q-Sayyoranorin perigeliy masofasi
2.Q-Sayyoranor afeliy masofasi
3.aq-Sayyora orbitasining katta yarim o'qini perigeliy masofasi orgali topish
4.aQ-Sayyora orbitasining katta yarim o'qini afeliy masofasi orgali topish
4
Sayyoranorin perigeliy masofasini kiriting.....
0.5
Sayyoranorin quyosh atrofida aylanish ekssentrisitetini kiriting.....
1
Sayyoranor orbita radiusi aQ= 0.25
*
*
*
*
*
*
Qaysi parametrn topmoqchisiz.....
1.a-Sayyora orbitasining katta yarim o'qi
2.V-Sayyoranor o'ttacha orbital tezligi
3.Perigeliy va afeliy masofalari
3
print(*, end="")
elif soz == "aq" or soz == "3":
    print("Sayyoranorin perigeliy masofasini kiriting....")
    q = float(input())
    print("Sayyoranorin quyosh atrofida aylanish ekssentrisitetini kiriting.... ")
    e = float(input())
    print("Sayyoranor orbita radiusi aq= ", q / (1 - e))
    for i in range(10):
        print("*", end="")
elif soz == "aQ" or soz == "4":
    print("Sayyoranorin afeliy masofasini kiriting....")
    Q = float(input())
    print("Sayyoranorin quyosh atrofida aylanish ekssentrisitetini kiriting.... ")
    e = float(input())
    print("Sayyoranor orbita radiusi aQ= ", Q / (1 + e))
    for i in range(10):

```

elif sozi == "aq" or sozi == "3":
 print("Sayyoranorin perigeliy masofasini kiriting....")
 q = float(input())
 print("Sayyoranorin quyosh atrofida aylanish ekssentrisitetini kiriting.... ")
 e = float(input())
 print("Sayyoranor orbita radiusi aq= ", q / (1 - e))
 for i in range(10):
 print("*", end="")
elif sozi == "aQ" or sozi == "4":
 print("Sayyoranorin afeliy masofasini kiriting....")
 Q = float(input())
 print("Sayyoranorin quyosh atrofida aylanish ekssentrisitetini kiriting.... ")
 e = float(input())
 print("Sayyoranor orbita radiusi aQ= ", Q / (1 + e))
 for i in range(10):

```
print("*", end="")
```

Animatsiya kodi:

```
import sys
import pygame
from pygame.locals import *
from OpenGL.GL import *
from OpenGL.GLU import *
import math
```

```
# Initialize Pygame and OpenGL
pygame.init()
width, height = 1000, 700
pygame.display.set_mode((width, height), DOUBLEBUF | OPENGL)
```

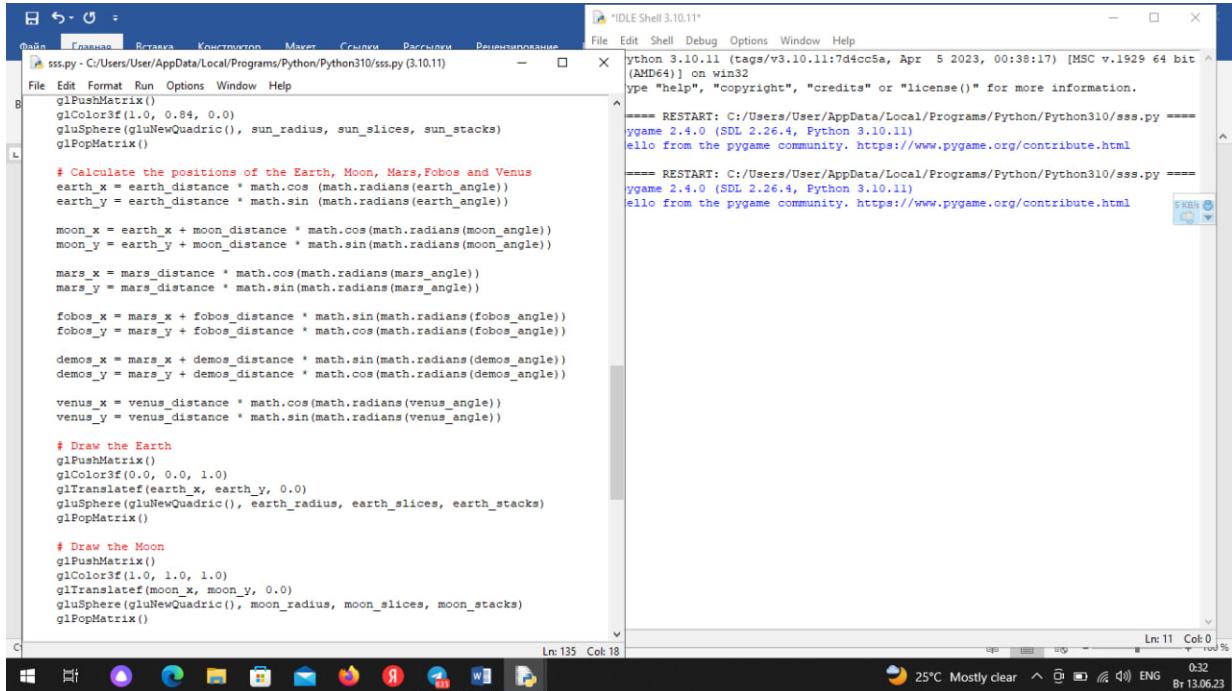
```
# Set up the projection matrix
glMatrixMode(GL_PROJECTION)
gluPerspective(45, (width / height), 0.1, 50.0)
```

```
# Set up the modelview matrix
glMatrixMode(GL_MODELVIEW)
gluLookAt(0, -20, 10, 0, 0, 0, 0, 0, 1)
```

```
# Enable depth testing
glEnable(GL_DEPTH_TEST)
```

Rasm-5:

ISSN (E): 2181-4570



```

sss.py - C:/Users/User/AppData/Local/Programs/Python/Python310/ssss.py (3.10.11)
File Edit Format Run Options Window Help
glPushMatrix()
glColor3f(1.0, 0.84, 0.0)
gluSphere(glUnNewQuadric(), sun_radius, sun_slices, sun_stacks)
glPopMatrix()

# Calculate the positions of the Earth, Moon, Mars, Phobos and Venus
earth_x = earth_distance * math.cos(math.radians(earth_angle))
earth_y = earth_distance * math.sin(math.radians(earth_angle))

moon_x = earth_x + moon_distance * math.cos(math.radians(moon_angle))
moon_y = earth_y + moon_distance * math.sin(math.radians(moon_angle))

mars_x = mars_distance * math.cos(math.radians(mars_angle))
mars_y = mars_distance * math.sin(math.radians(mars_angle))

phobos_x = mars_x + phobos_distance * math.sin(math.radians(phobos_angle))
phobos_y = mars_y + phobos_distance * math.cos(math.radians(phobos_angle))

demos_x = mars_x + demos_distance * math.sin(math.radians(demos_angle))
demos_y = mars_y + demos_distance * math.cos(math.radians(demos_angle))

venus_x = venus_distance * math.cos(math.radians(venus_angle))
venus_y = venus_distance * math.sin(math.radians(venus_angle))

# Draw the Earth
glPushMatrix()
glColor3f(0.0, 0.0, 1.0)
glTranslatef(earth_x, earth_y, 0.0)
gluSphere(glUnNewQuadric(), earth_radius, earth_slices, earth_stacks)
glPopMatrix()

# Draw the Moon
glPushMatrix()
glColor3f(1.0, 1.0, 1.0)
glTranslatef(moon_x, moon_y, 0.0)
gluSphere(glUnNewQuadric(), moon_radius, moon_slices, moon_stacks)
glPopMatrix()

```

File Edit Shell Debug Options Window Help

python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr 5 2023, 00:38:17) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

===== RESTART: C:/Users/User/AppData/Local/Programs/Python/Python310/ssss.py =====
pygame 2.4.0 (SDL 2.26.4, Python 3.10.11)
Hello from the pygame community. https://www.pygame.org/contribute.html

===== RESTART: C:/Users/User/AppData/Local/Programs/Python/Python310/ssss.py =====
pygame 2.4.0 (SDL 2.26.4, Python 3.10.11)
Hello from the pygame community. https://www.pygame.org/contribute.html

Set up the Sun

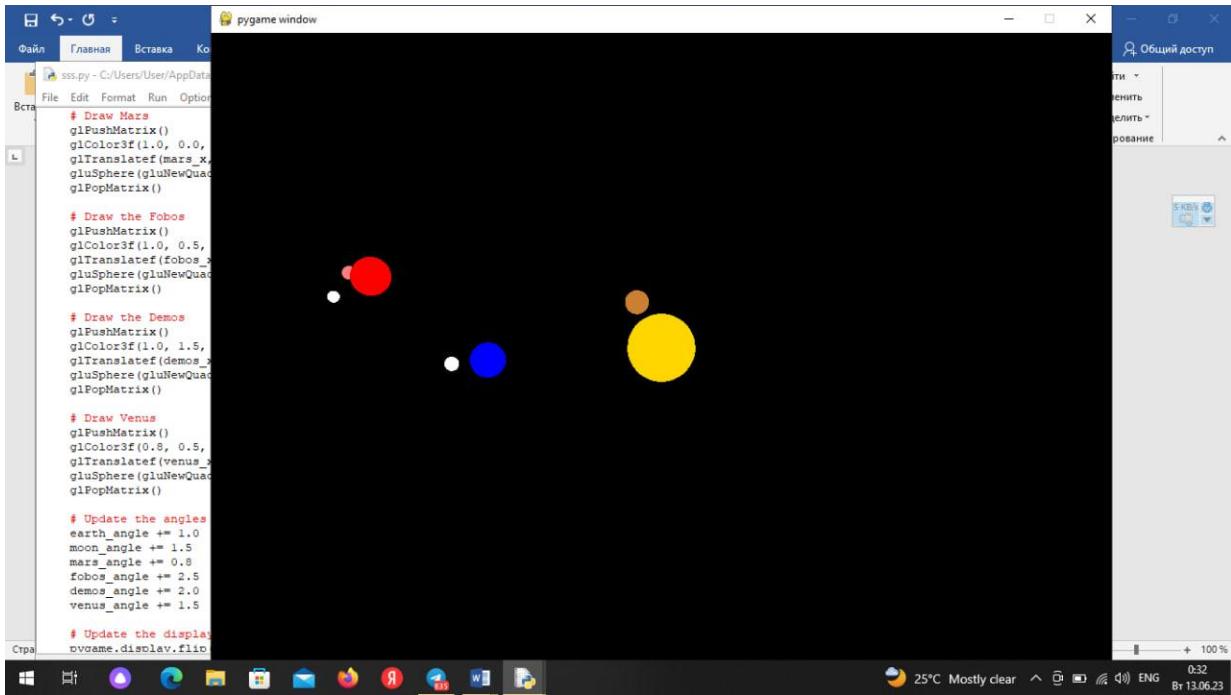
sun_radius = 1.0

sun_slices = 30

sun_stacks = 30

Rasm-6:

ISSN (E): 2181-4570



```

# Set up the Earth
earth_radius = 0.5
earth_distance = 5.0
earth_slices = 30
earth_stacks = 30
earth_angle = 0.0

```

```

# Set up the Moon
moon_radius = 0.2
moon_distance = 1.0
moon_slices = 30
moon_stacks = 30
moon_angle = 90.0

```

```

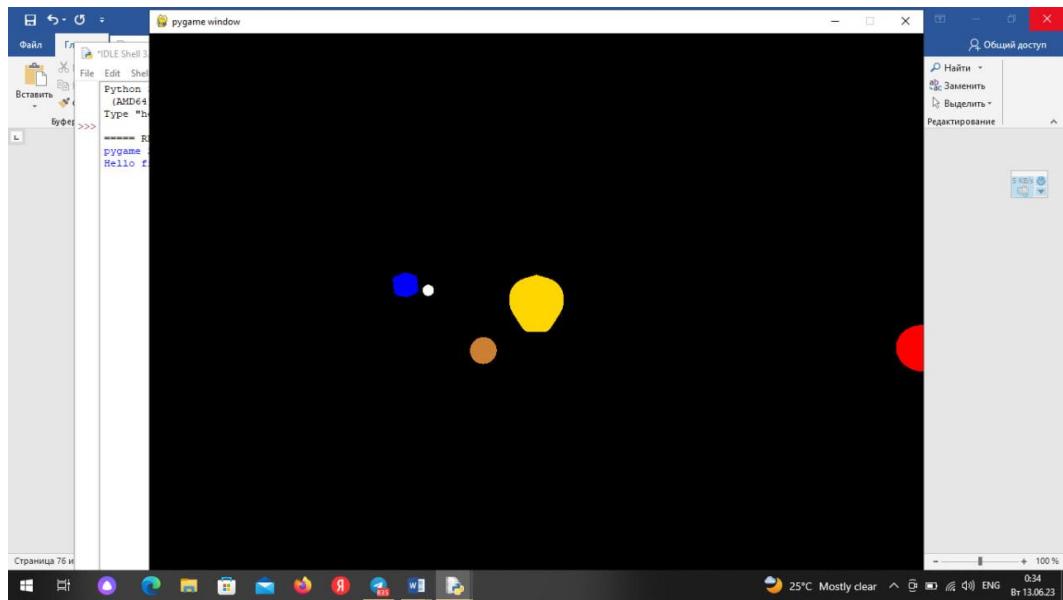
# Set up Mars
mars_radius = 0.7
mars_distance = 12.0
mars_slices = 30
mars_stacks = 30

```

ISSN (E): 2181-4570

`mars_angle = 0.0`

Rasm-7:



`# Set up the Fobos`

`fobos_radius = 0.25`

`fobos_distance = 1.0`

`fobos_slices = 30`

`fobos_stacks = 30`

`fobos_angle = 0.0`

`# Set up the Demos`

`demos_radius = 0.2`

`demos_distance = 2.0`

`demos_slices = 30`

`demos_stacks = 30`

`demos_angle = 180.0`

`# Set up Venus`

`venus_radius = 0.4`

`venus_distance = 3.5`

`venus_slices = 30`

ISSN (E): 2181-4570

venus_stacks = 30

venus_angle = 0.0

Main animation loop

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

Clear the screen and depth buffer

glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)

Draw the Sun

glPushMatrix()

glColor3f(1.0, 0.84, 0.0)

gluSphere(gluNewQuadric(), sun_radius, sun_slices, sun_stacks)

glPopMatrix()

Calculate the positions of the Earth, Moon, Mars,Fobos and Venus

earth_x = earth_distance * math.cos (math.radians(earth_angle))

earth_y = earth_distance * math.sin (math.radians(earth_angle))

moon_x = earth_x + moon_distance * math.cos(math.radians(moon_angle))

moon_y = earth_y + moon_distance * math.sin(math.radians(moon_angle))

mars_x = mars_distance * math.cos(math.radians(mars_angle))

mars_y = mars_distance * math.sin(math.radians(mars_angle))

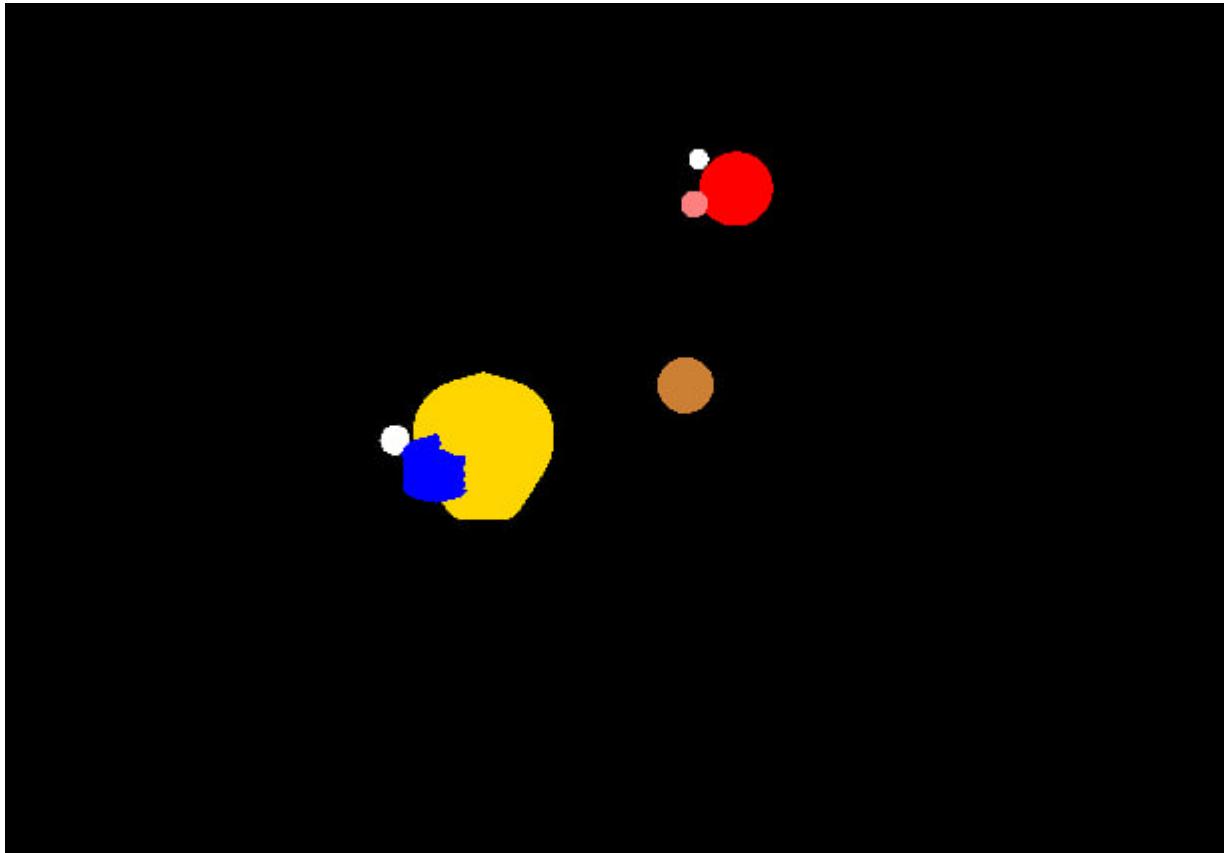
fobos_x = mars_x + fobos_distance * math.sin(math.radians(fobos_angle))

fobos_y = mars_y + fobos_distance * math.cos(math.radians(fobos_angle))

demos_x = mars_x + demos_distance * math.sin(math.radians(demos_angle))

ISSN (E): 2181-4570

demos_y = mars_y + demos_distance * math.cos(math.radians(demos_angle))
Rasm-8:



venus_x = venus_distance * math.cos(math.radians(venus_angle))
venus_y = venus_distance * math.sin(math.radians(venus_angle))

```
# Draw the Earth
glPushMatrix()
glColor3f(0.0, 0.0, 1.0)
glTranslatef(earth_x, earth_y, 0.0)
gluSphere(gluNewQuadric(), earth_radius, earth_slices, earth_stacks)
glPopMatrix()
```

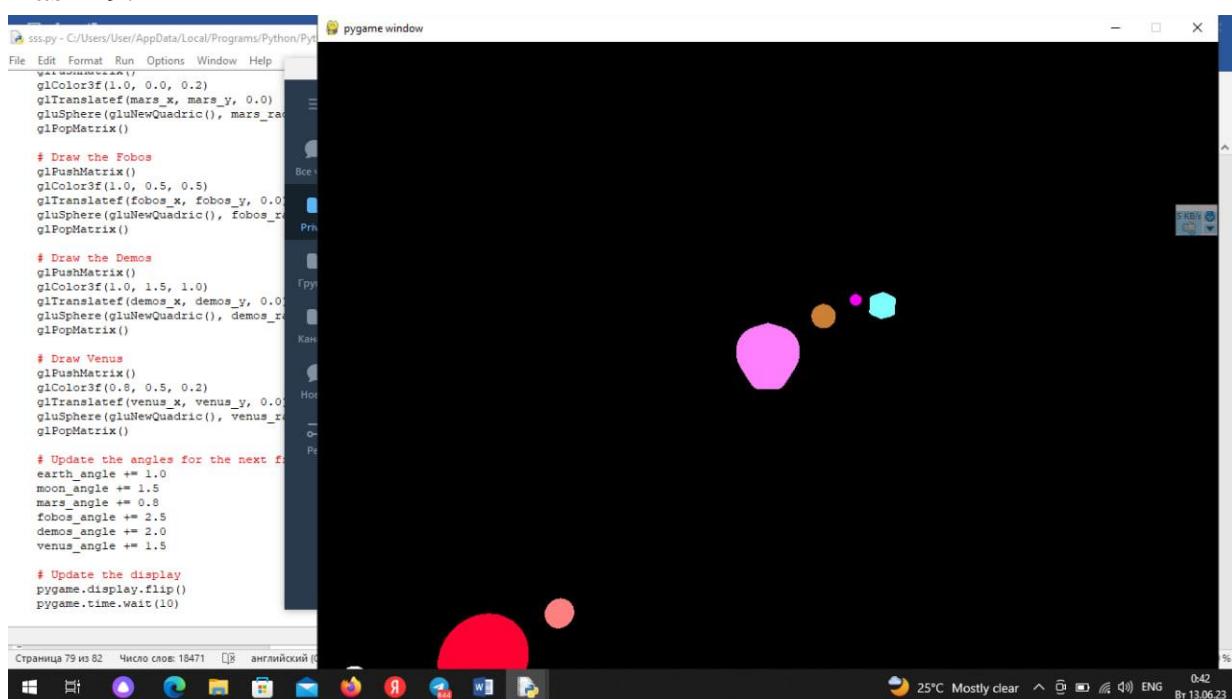
```
# Draw the Moon
glPushMatrix()
```

ISSN (E): 2181-4570

```
glColor3f(1.0, 1.0, 1.0)
glTranslatef(moon_x, moon_y, 0.0)
gluSphere(gluNewQuadric(), moon_radius, moon_slices, moon_stacks)
glPopMatrix()
```

```
# Draw Mars
glPushMatrix()
glColor3f(1.0, 0.0, 0.0)
glTranslatef(mars_x, mars_y, 0.0)
gluSphere(gluNewQuadric(), mars_radius, mars_slices, mars_stacks)
glPopMatrix()
```

Rasm-9:



```
# Draw the Fobos
glPushMatrix()
glColor3f(1.0, 0.5, 0.5)
glTranslatef(fobos_x, fobos_y, 0.0)
gluSphere(gluNewQuadric(), fobos_radius, fobos_slices, fobos_stacks)
glPopMatrix()
```

Draw the Demos

```
glPushMatrix()  
glColor3f(1.0, 1.5, 1.0)  
glTranslatef(demos_x, demos_y, 0.0)  
gluSphere(gluNewQuadric(), demos_radius, demos_slices, demos_stacks)  
glPopMatrix()
```

Draw Venus

```
glPushMatrix()  
glColor3f(0.8, 0.5, 0.2)  
glTranslatef(venus_x, venus_y, 0.0)  
gluSphere(gluNewQuadric(), venus_radius, venus_slices, venus_stacks)  
glPopMatrix()
```

Update the angles for the next frame

```
earth_angle += 1.0  
moon_angle += 1.5  
mars_angle += 0.8  
fobos_angle += 2.5  
demos_angle += 2.0  
venus_angle += 1.5
```

Update the display

```
pygame.display.flip()  
pygame.time.wait(10)
```