Sprawozdanie

Celem zajęć było utworzenie programu, który będzie symulował oddziaływania grawitacyjne dwóch lub więcej ciał.

Pisanie programu zaczęto od zaimportowania wymaganych bibliotek:

```
1   import pygame
2   import numpy as np
```

Następnie utworzono wartość stałą grawitacji oraz klasę, która definiuję narysowany punkt:

```
4   g = 6.647 * 10**(-11)
5
6   class circle:
7       def __init__(self, xc, yc, r, color):
8           self.xc=xc
9           self.yc=yc
10          self.color=color
11          self.r = r
12      def draw(self,window):
13          pygame.draw.circle(window, self.color,
14          [self.xc, self.yc], self.r
15          )
16
```

Klasy tej następnie użyto aby utworzyć klasę child masspoint, odpowiedzialną za przypisanie masy do obiektu.

```
class massObject(circle):
    def __init__(self,xc,yc, r, color, mass, v):
        self.r=r
        self.xc=xc
        self.yc=yc
        self.color = color
        self.mass = mass
        self.v = v

        circle.__init__(self,xc,yc,self.r,color)
```

W klasie tej umieszczono metodę SimulateGravity odpowiedzialną za symulowanie grawitacji pomiędzy tym punktem a innym wybranym:

```
27      def SimulateGravity(self, obj, dt):
28          #define lists that will become a vector
29          templst = [self.xc, self.yc]
30          templst2 = [obj.xc, obj.yc]
31          #define a numpy vector
32          vctr = np.array(templst)
33          vctrobj = np.array(templst2)
34          #magnitude of the difference of vectors
35          mag = np.linalg.norm(vctrobj-vctr)
36
37          self.a = g*(obj.mass/((mag)**3))*((vctrobj-vctr))
38
39          self.v += self.a
40          #scaling of v
41          #if self.v[0] >= 3 or self.v[1] >= 3:
42          #    self.v *= 0.5
43
44
45          self.xc += self.v[0]
46          self.yc += self.v[1]
47
```

Następnie przystąpiono do pisania głównej pętli symulacji. Zaczęto od początkowych definicji:

```
51  pygame.init()
52  screen = pygame.display.set_mode((1280, 720))
53  clock = pygame.time.Clock()
54  running = True
55  objekt = massObject(680, 320, 5, "blue", 200000000000000, [0.1,0.1])
56  objekt2 = massObject(200, 200, 5, "red", 1000000000000,[1,-1])
```

A następnie napisano następujące polecenia w pętli odpowiedzialne za symulację:

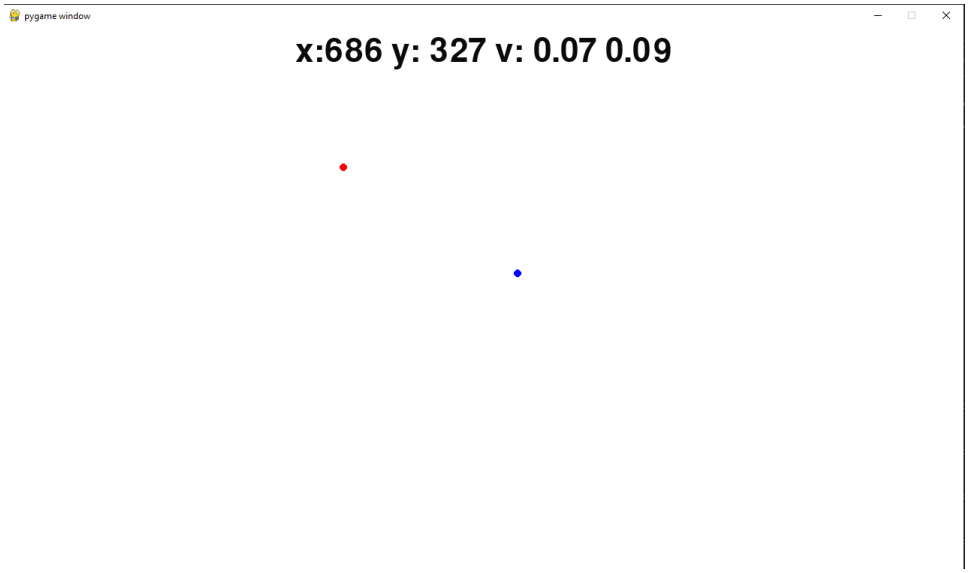
```
58  while running:
59      screen.fill("white")
60      objekt.draw(screen)
61      objekt2.draw(screen)
62      for event in pygame.event.get():
63          if event.type == pygame.QUIT:
64              running = False
65
66      objekt.SimulateGravity(objekt2, dt)
67      objekt2.SimulateGravity(objekt, dt)
```

Aby łatwiej debuggować program, napisano również kod odpowiedzialny za wyświetlanie wartości obiektu pierwszego:

Zakończono następującymi poleceniami:

```
74      pygame.display.flip()
75      dt = clock.tick(60) / 1000
76  pygame.quit()
77
```

Efekt wygląda następująco:



Gdzie kulki odpowiadają punktom masowym.