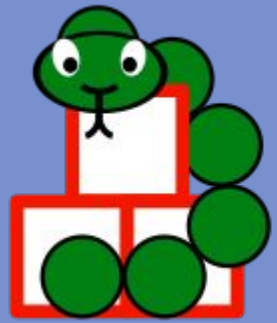
Pymunk - biblioteka fizyczna ułatwiająca tworzenie gier 2D i symulacji rzeczywistości z PyGame

Grzegorz Gołyś


pymunk

matplotlib



pymunk


kivy


matplotlib


pymunk


kivy
matplotlib
pymunk

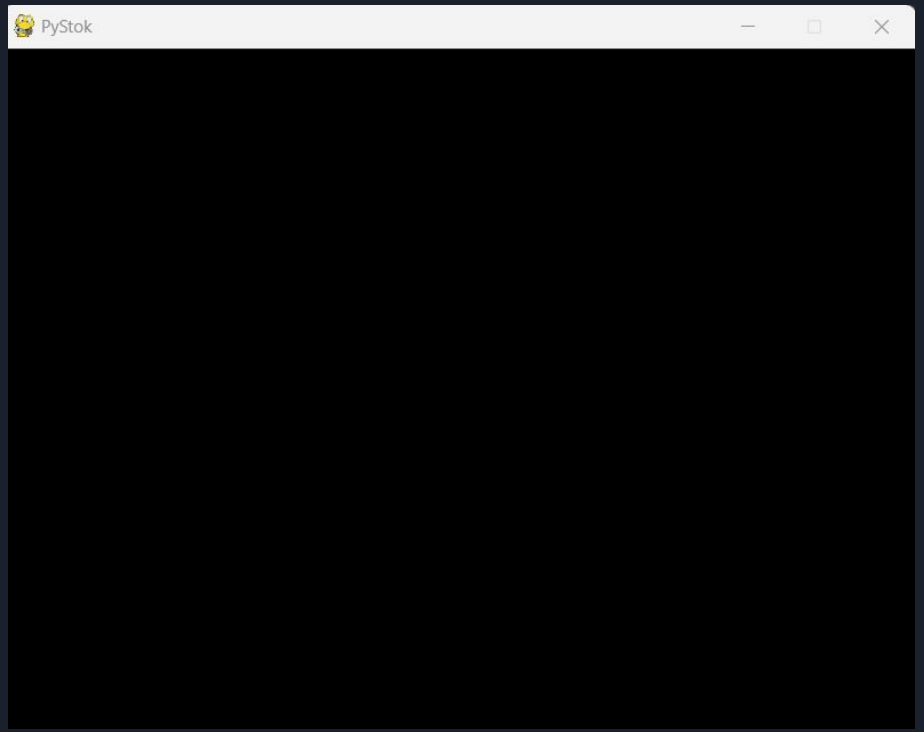

kivy
matplotlib
pymunk
pygame


pygame


pymunk

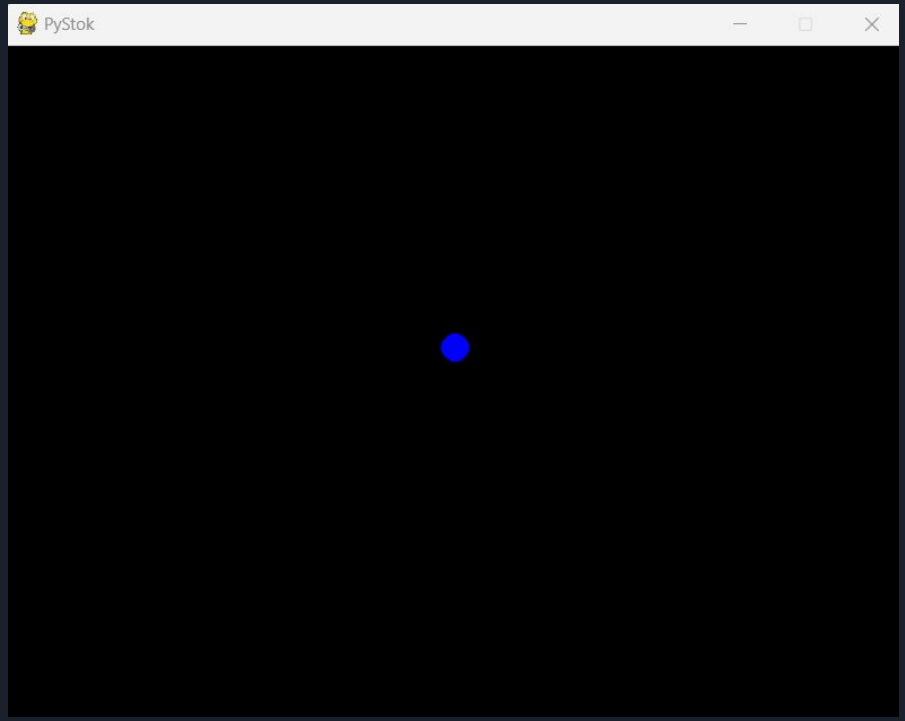
pygame

```python
import pygame
pygame.init()
scena = pygame.display.set_mode((640, 480))
pygame.display.set_caption("PyStok")
run = True
while run:
    scena.fill((0, 0, 0))
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
            break
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                run = False
                break
    pygame.display.update()
pygame.quit()
```



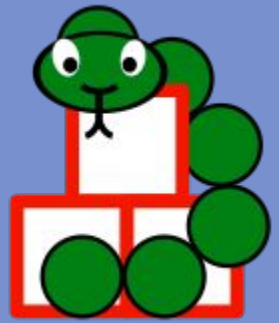
pygame

```python
import pygame
pygame.init()
scena = pygame.display.set_mode((640, 480))
pygame.display.set_caption("PyStok")
run = True
x, y = scena.get_width()/2, 0
t = pygame.time.Clock()
while run:
    scena.fill((0, 0, 0))
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False
            break
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                run = False
                break
    pygame.draw.circle(scena, (0, 0, 255), (x, y), 10)
    y += 1
    pygame.display.update()
    t.tick(60)
pygame.quit()
```



The Pymunk Vision :

“Make 2d physics easy to include in your game”

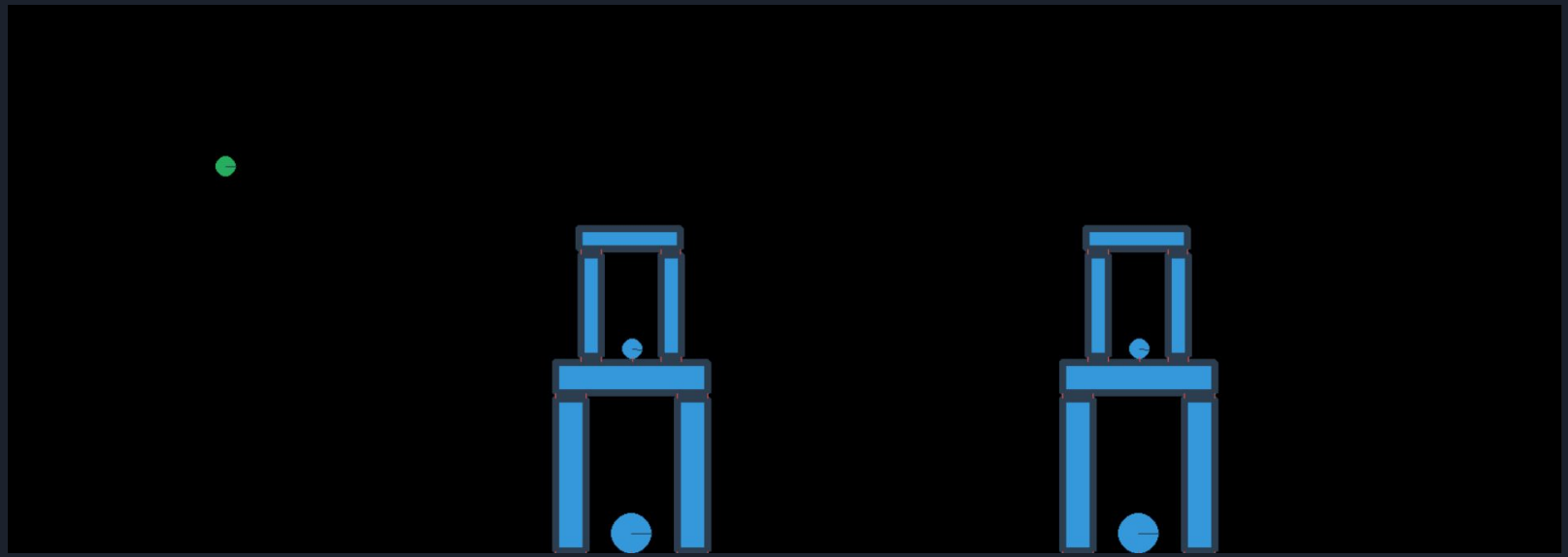


pymunk

```
space = pymunk.Space()
space.gravity = 0, 981
option = pymunk.pygame_util.DrawOptions(scena)
```

```
space.debug_draw(option)
space.step(1/fps)
```

Angry birds



Funkcja tworząca piłke

```
def create_ball(pos, radius, mass, sfera, collision_type=1):
    moment = pymunk.moment_for_circle(mass, 0, radius)
    b = pymunk.Body(mass, moment, body_type=pymunk.Body.KINEMATIC)
    b.position = pos
    circ = pymunk.Circle(b, radius)
    circ.friction = 0.3
    circ.elasticity = 0.1
    circ.collision_type = collision_type
    sfera.add(b, circ)
    return circ
```

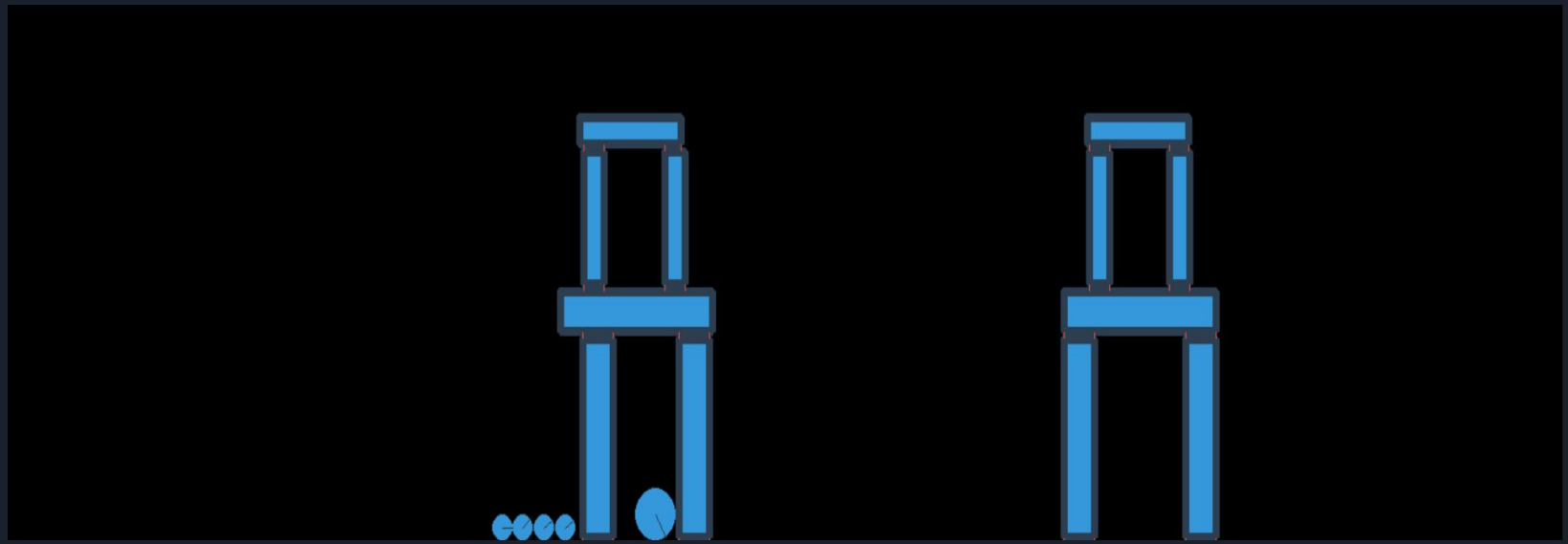
Funkcja tworząca budynki

```
def create_home(mass, fri, elast, col_type, sfera, szerokosc, wysokosc, *pos):
    """mass odpowiedzialny za mase obiektów, fri - to friction, elast
    elastycznosc obiektow, col_type - typ kolizji,sfera to space, pos to pozycja trzech elementow"""
    size = [(szerokosc, wysokosc), (szerokosc, wysokosc), (wysokosc, szerokosc)]
    bl = pymunk.Body(mass)
    bp = pymunk.Body(mass)
    bd = pymunk.Body(mass)
    body = [bl, bp, bd]
    shape = []
    for position, bod, siz in zip(pos, body, size):
        bod.position = position
        pol = pymunk.Poly.create_box(bod, siz, 3)
        pol.friction = fri
        pol.elasticity = elast
        pol.mass = mass
        pol.collision_type = col_type
        shape.append(pol)
    sfera.add(*body, *shape)
    return shape
```

Funkcja tworząca wielokąt

```
def create_poly(mass, friction, elasticity, position, collide_type, sfera, *cor):
    moment = pymunk.moment_for_poly(mass, cor)
    body = pymunk.Body(mass, moment)
    body.position = position
    poly = pymunk.Poly(body, cor)
    poly.friction = friction
    poly.elasticity = elasticity
    poly.collision_type = collide_type
    sfera.add(body, poly)
    return poly
```

Obsługa kolizji



Obsługa kolizji

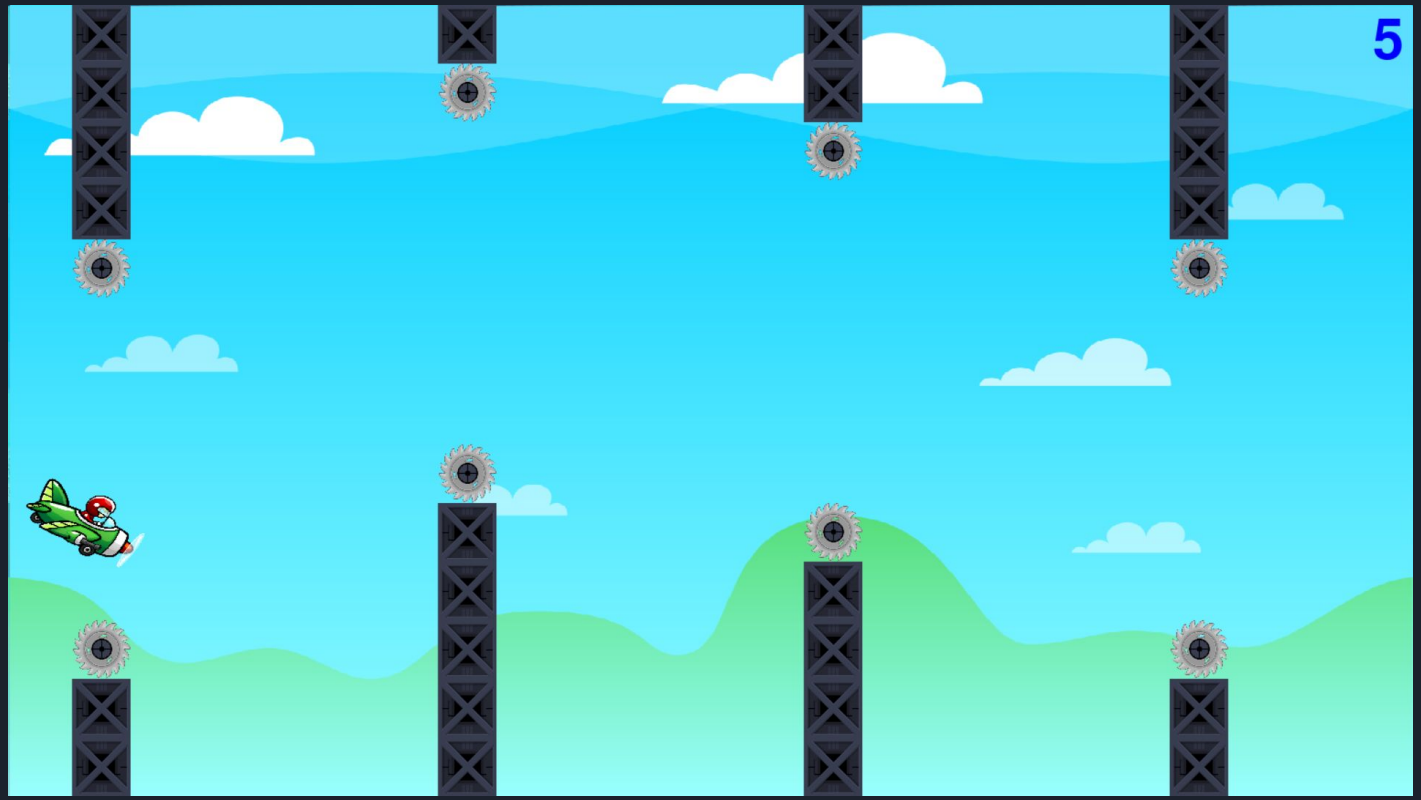
```
k_2_3 = space.add_collision_handler(2, 3)
k_2_3.begin = begin
k_2_3.pre_solve = set_features
k_2_3.post_solve = info
k_2_3.separate = separate
```

```
def begin(arbiter, space, data):
    print("kolizja")
    return True


def set_features(arbiter, space, data):
    arbiter.friction = 1
    arbiter.elasticity = 0.3
    return True


def info(arbiter, space, data):
    print(arbiter.total_ke)
    if arbiter.total_ke > 300000:
        space.remove(arbiter.shapes[0].body, arbiter.shapes[0])
    return True


def separate(arbiter, space, data):
    print("rozlaczyly sie")
```

Impuls

```
if event.type == pygame.MOUSEBUTTONDOWN:
    x, y = pygame.mouse.get_pos()
    rad = math.atan2(ball.body.position[1] - y, ball.body.position[0] - x)
    ball.body.body_type = pymunk.Body.DYNAMIC
    ball.mass = 50
    x = math.cos(rad)
    y = math.sin(rad)
    ball.body.apply_impulse_at_local_point((x * 70000, y * 70000), (0, 0))
```

Bird - Plane

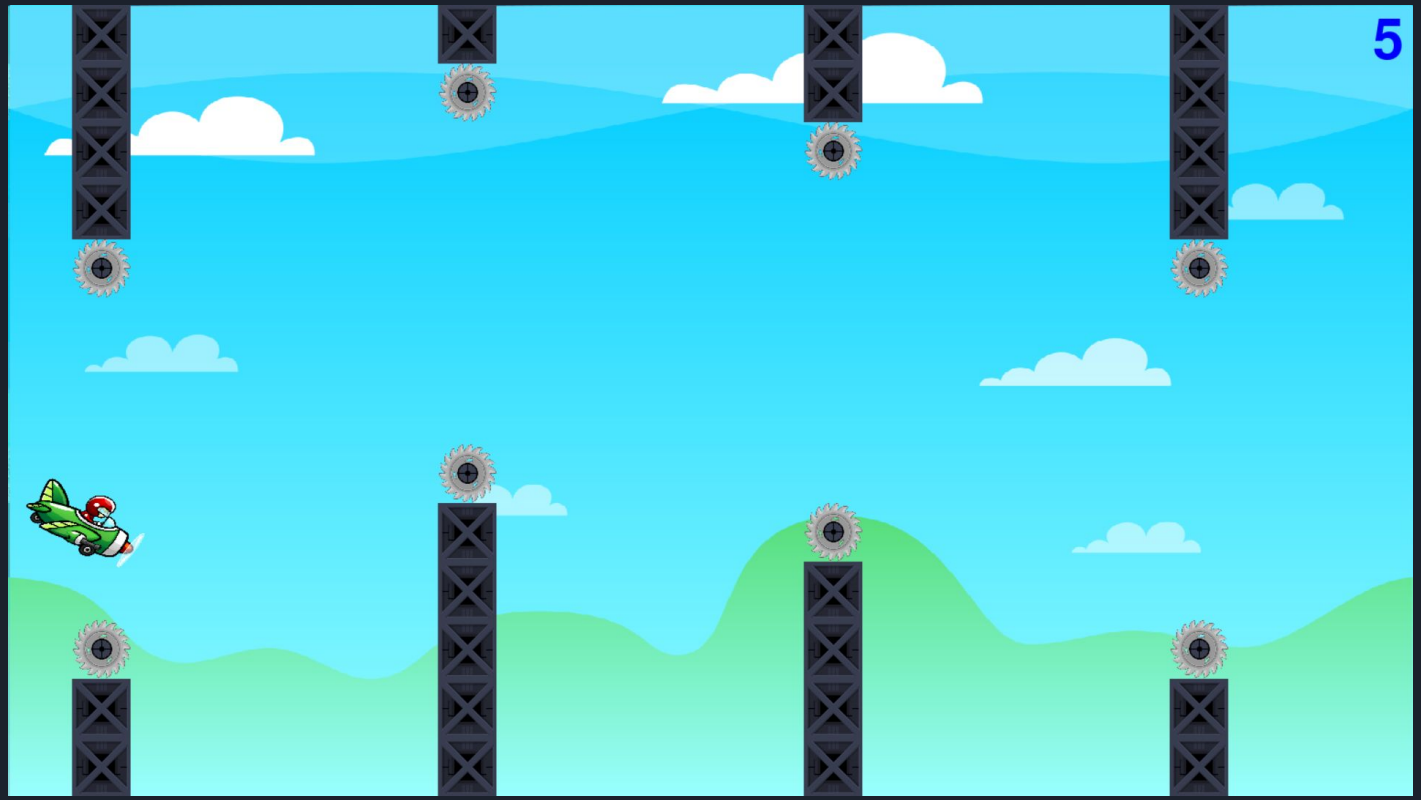


Klasa Plane

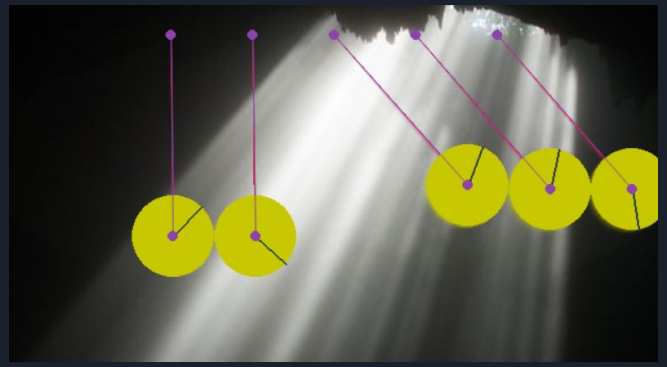
```
class Plane(pygame.sprite.Sprite):
    def __init__(self, shape, pictures):
        super().__init__()
        self.i = 0
        self.pictures = pictures
        self.shape = shape
        self.image = self.pictures[int(self.i)]
        self.rect = self.image.get_rect()
        self.rect.center = self.shape.body.center

    def update(self):
        self.i += 0.15
        if self.i >= len(self.pictures):
            self.i = 0
        self.image = self.pictures[int(self.i)]
        angle = 30 if self.shape.body.velocity[1] < 0 else -30
        self.image = pygame.transform.rotate(self.image, angle)
        self.rect.center = self.shape.body.position
```

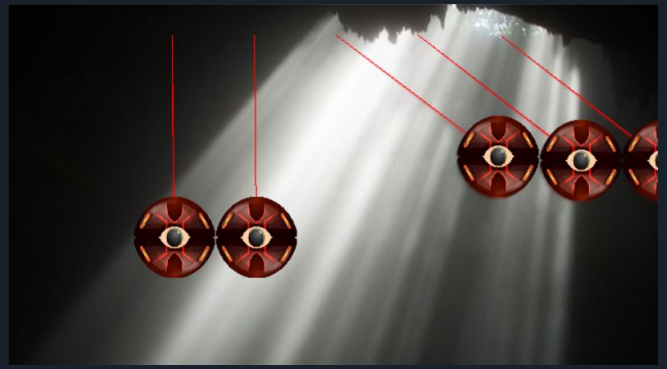
```
pygame.sprite.spritecollide(plane, cubes, False, pygame.sprite.collide_mask)
pygame.sprite.spritecollideany(plane, cubes, pygame.sprite.collide_mask)
pygame.sprite.groupcollide(planes, cubes, False, False, pygame.sprite.collide_mask)
pygame.sprite.groupcollide(planes, cubes, False, False, pygame.sprite.collide_rect)
```

Bird - Plane

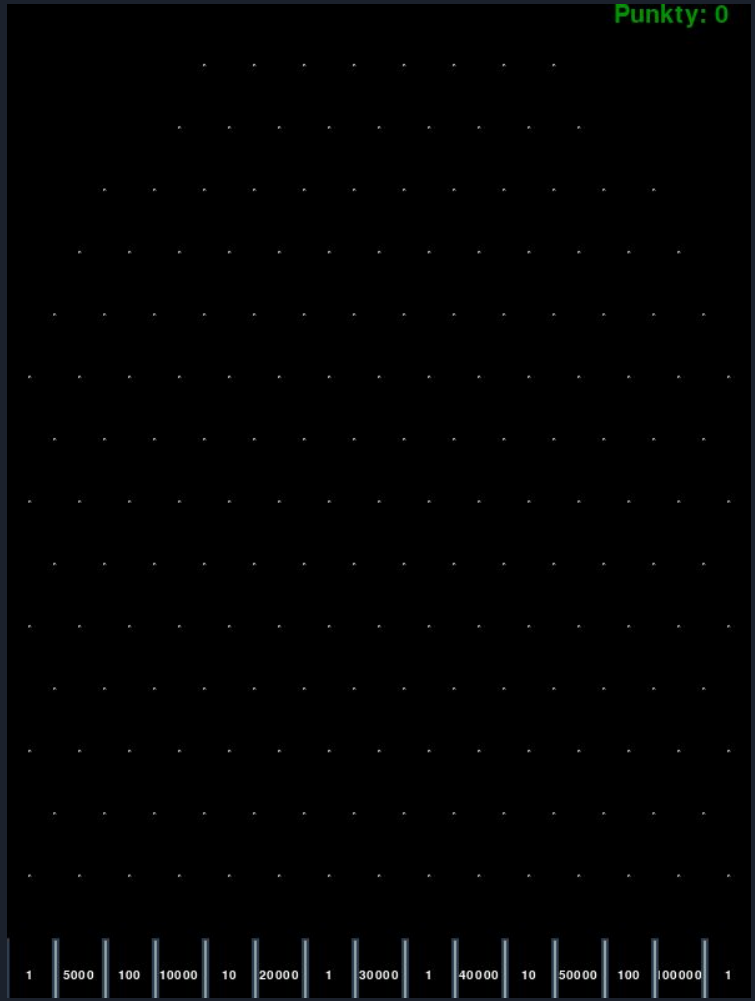


Wahadło Newtona

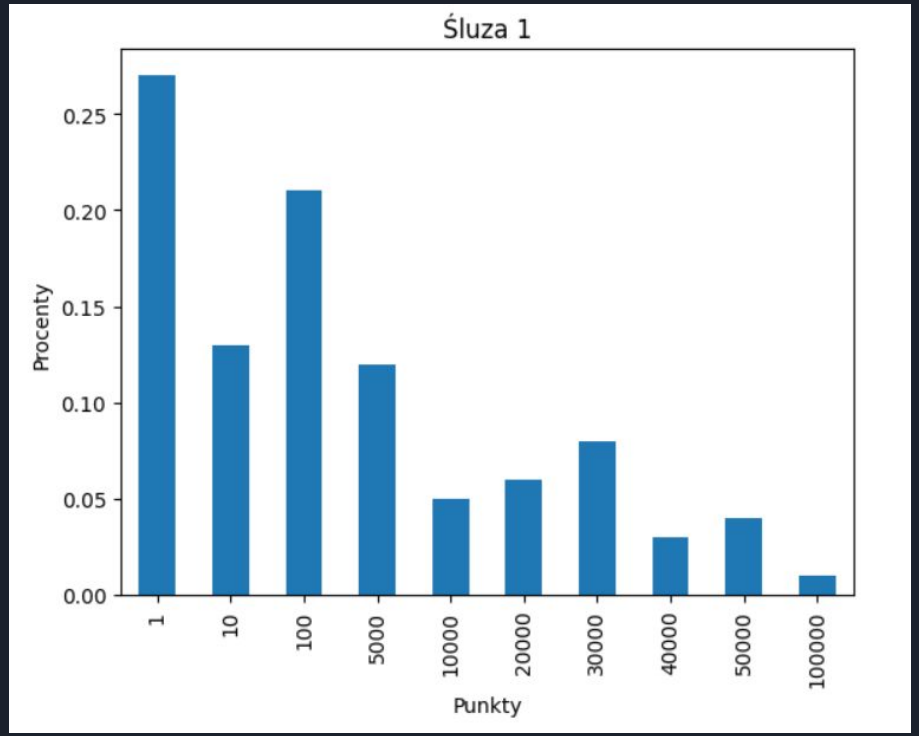




The Wall. Wygraj marzenia
Gra z rekordową wygraną

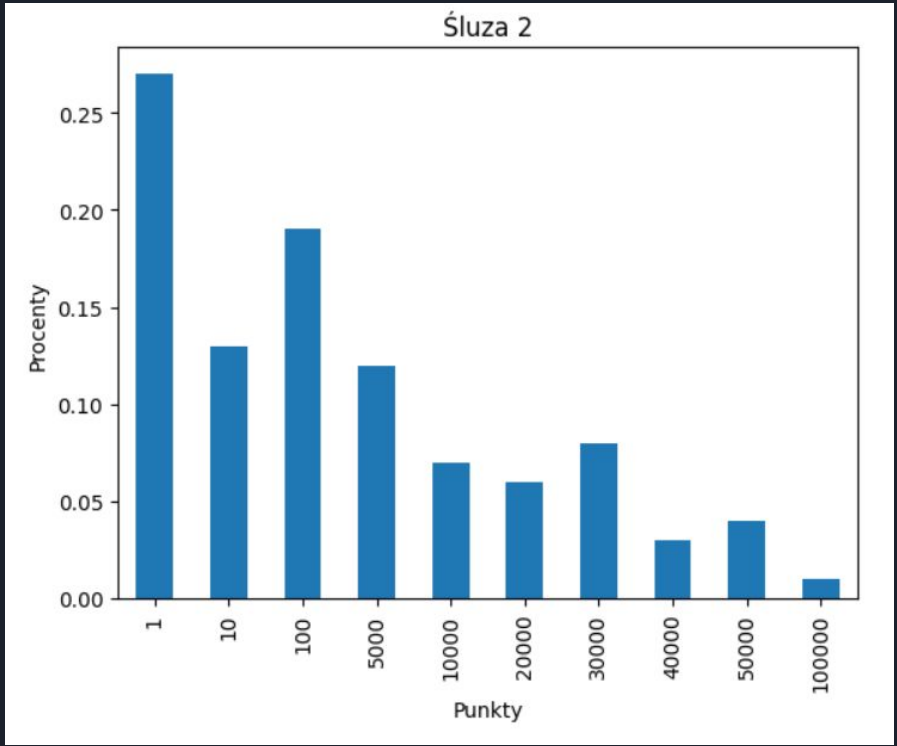

Punkty: 0
1
5000
100
10000
10
20000
1
30000
1
40000
10
50000
100
100000
1

Wartość oczekiwana : 8 922.57



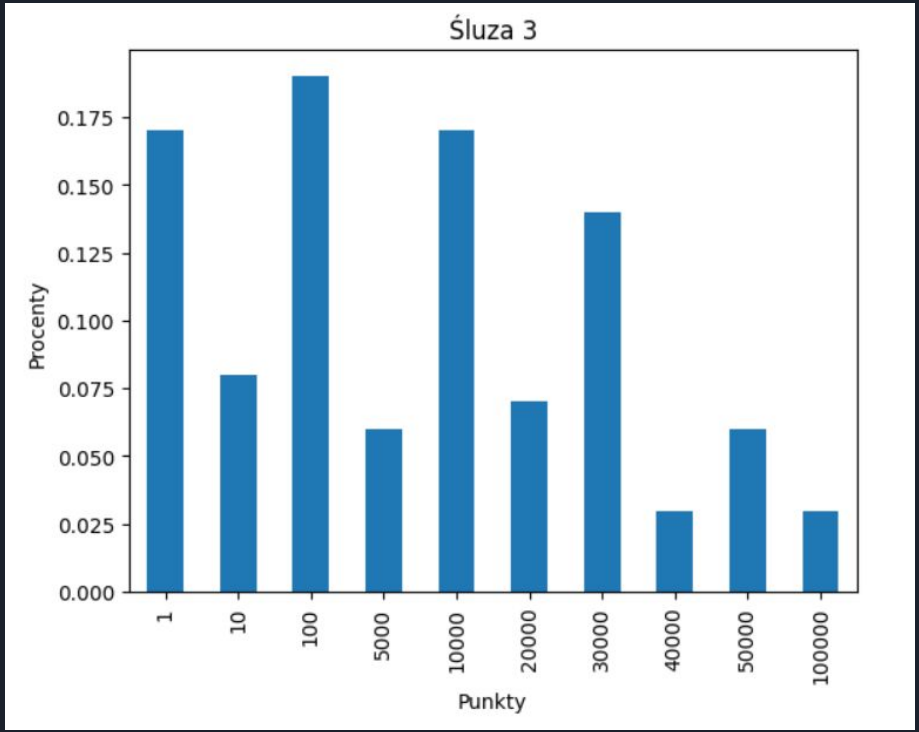
pkt	
1	0.27
10	0.13
100	0.21
5000	0.12
10000	0.05
20000	0.06
30000	0.08
40000	0.03
50000	0.04
100000	0.01

Wartość oczekiwana : 9 120.57



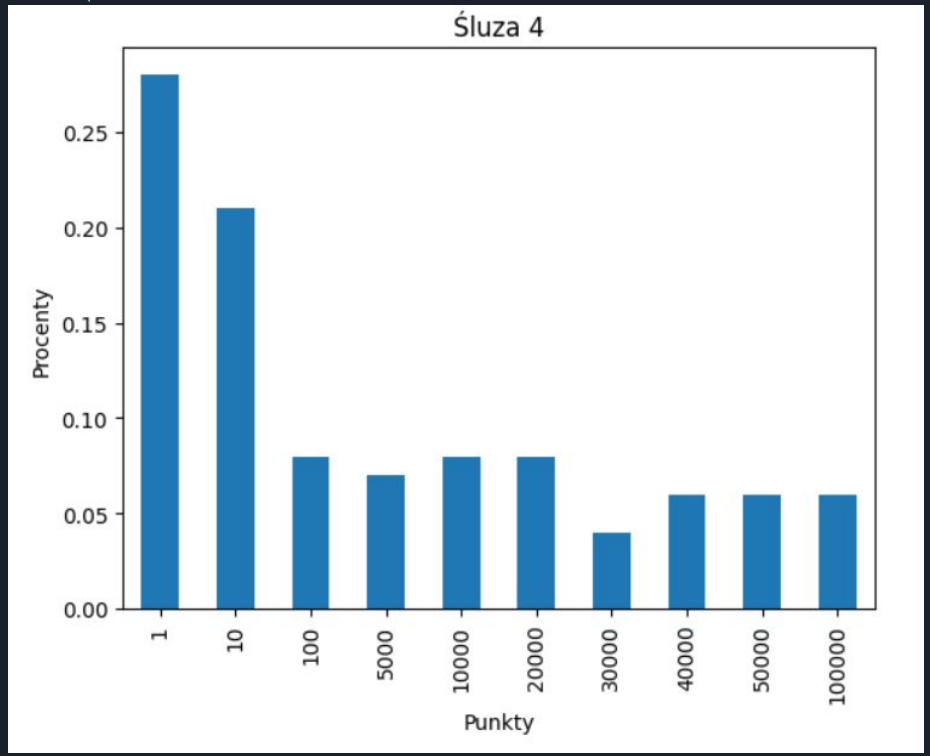
pkt	
1	0.27
10	0.13
100	0.19
5000	0.12
10000	0.07
20000	0.06
30000	0.08
40000	0.03
50000	0.04
100000	0.01

Wartość oczekiwana : 14 819.97



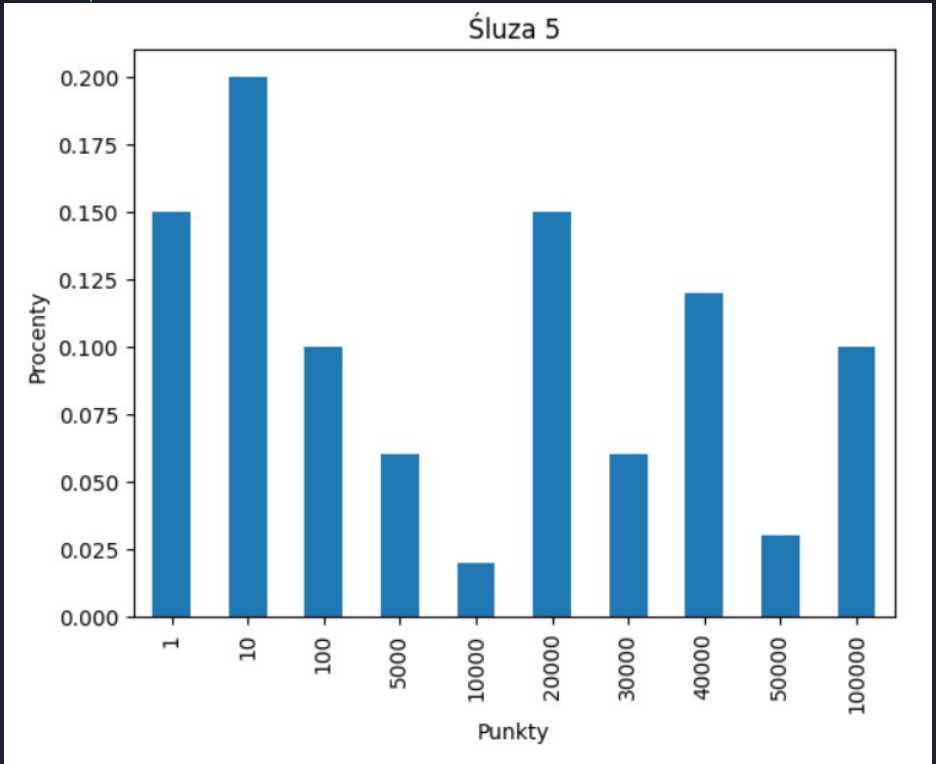
pkt	
1	0.17
10	0.08
100	0.19
5000	0.06
10000	0.17
20000	0.07
30000	0.14
40000	0.03
50000	0.06
100000	0.03

Wartość oczekiwana : 15 360.38



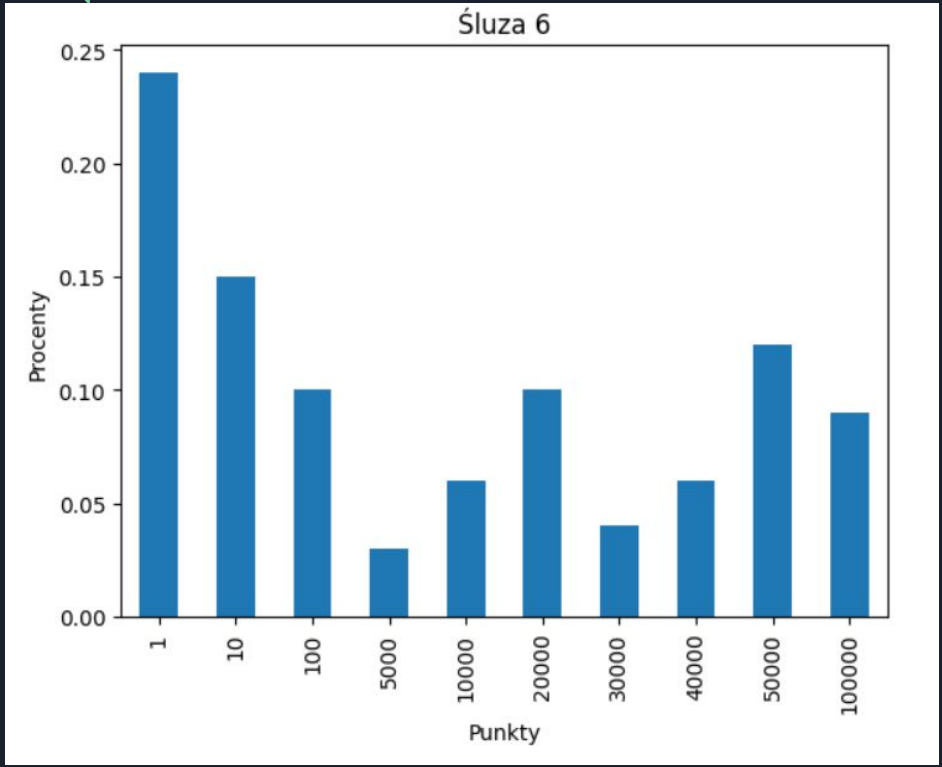
pkt	
1	0.28
10	0.21
100	0.08
5000	0.07
10000	0.08
20000	0.08
30000	0.04
40000	0.06
50000	0.06
100000	0.06

Wartość oczekiwana : 21 612.15



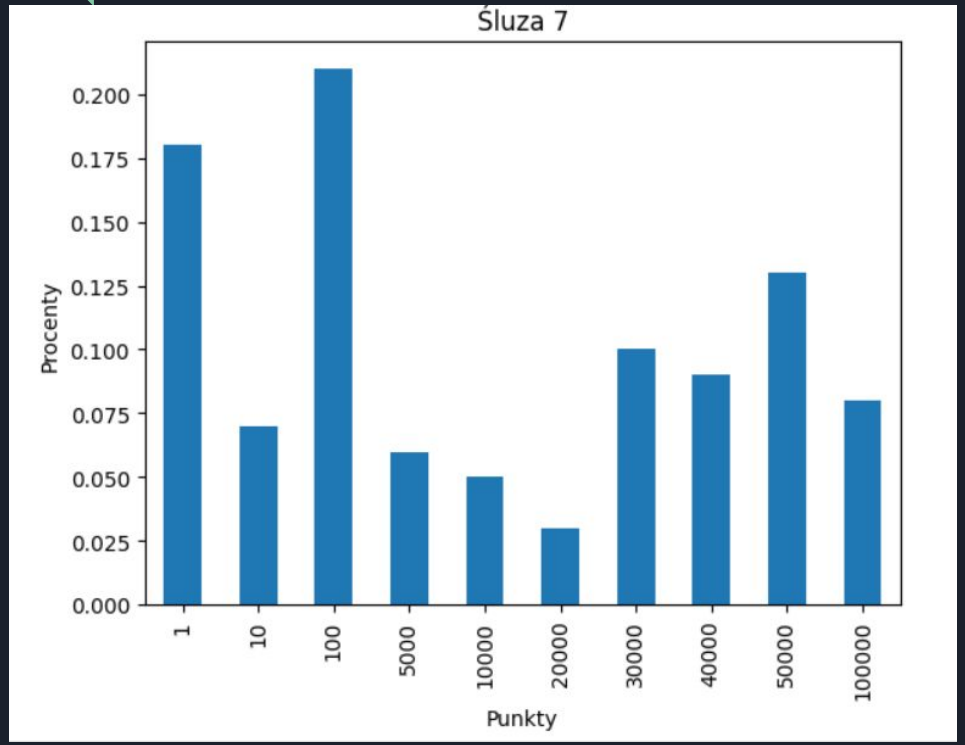
pkt	
1	0.15
10	0.20
100	0.10
5000	0.06
10000	0.02
20000	0.15
30000	0.06
40000	0.12
50000	0.03
100000	0.10

Wartość oczekiwana : 21 361.74

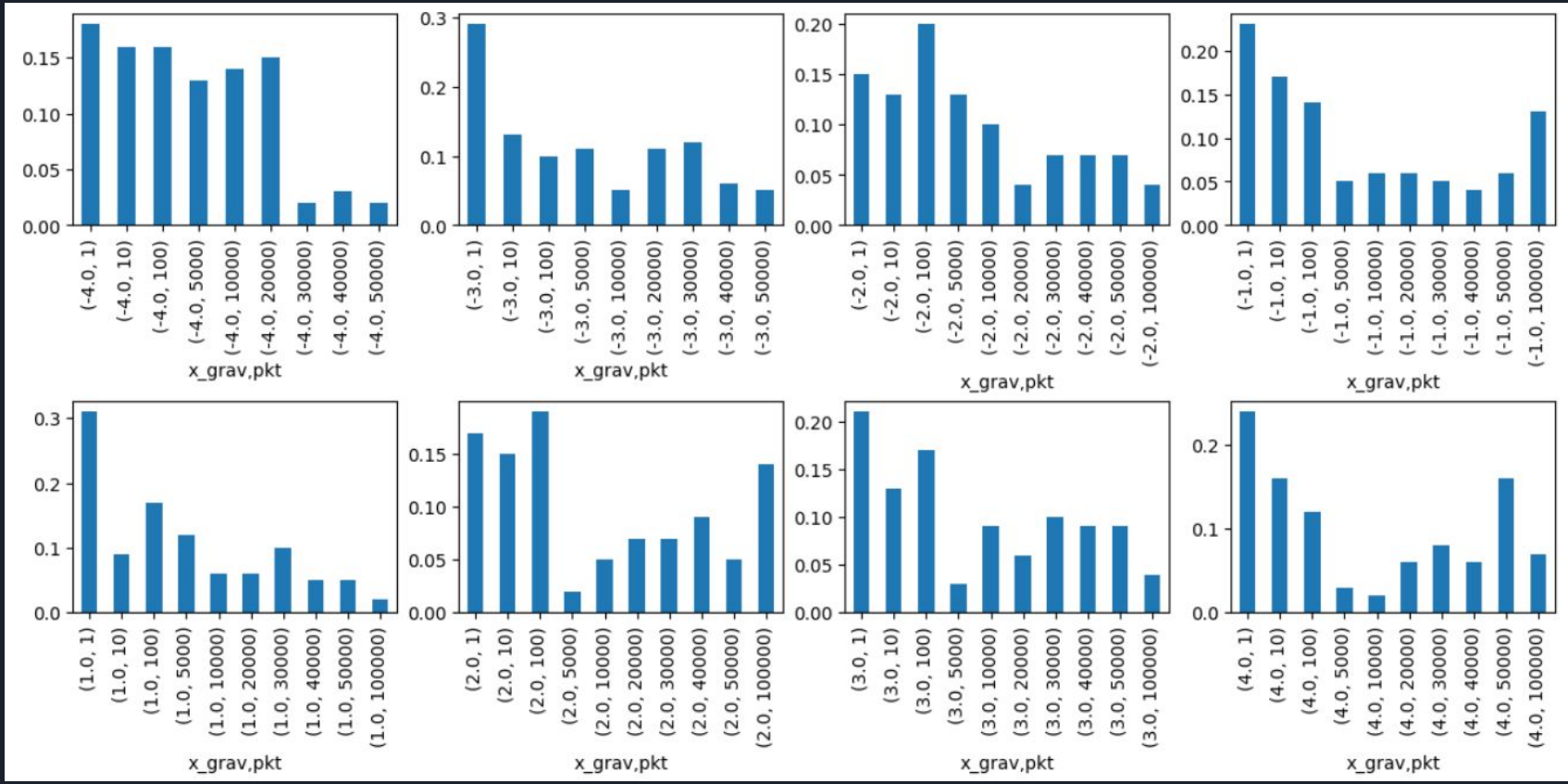


pkt	
1	0.24
10	0.15
100	0.10
5000	0.03
10000	0.06
20000	0.10
30000	0.04
40000	0.06
50000	0.12
100000	0.09

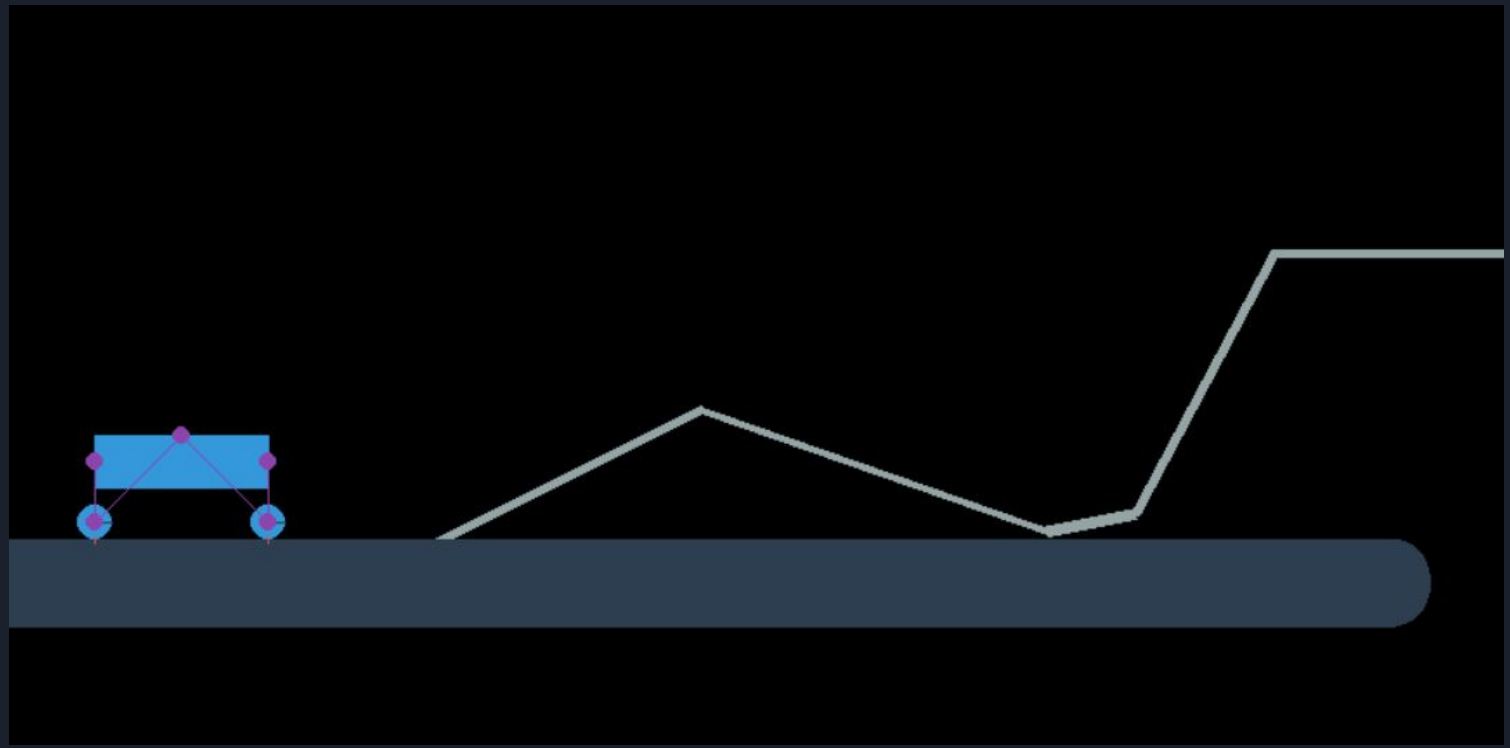
Wartość oczekiwana : 20 421.88

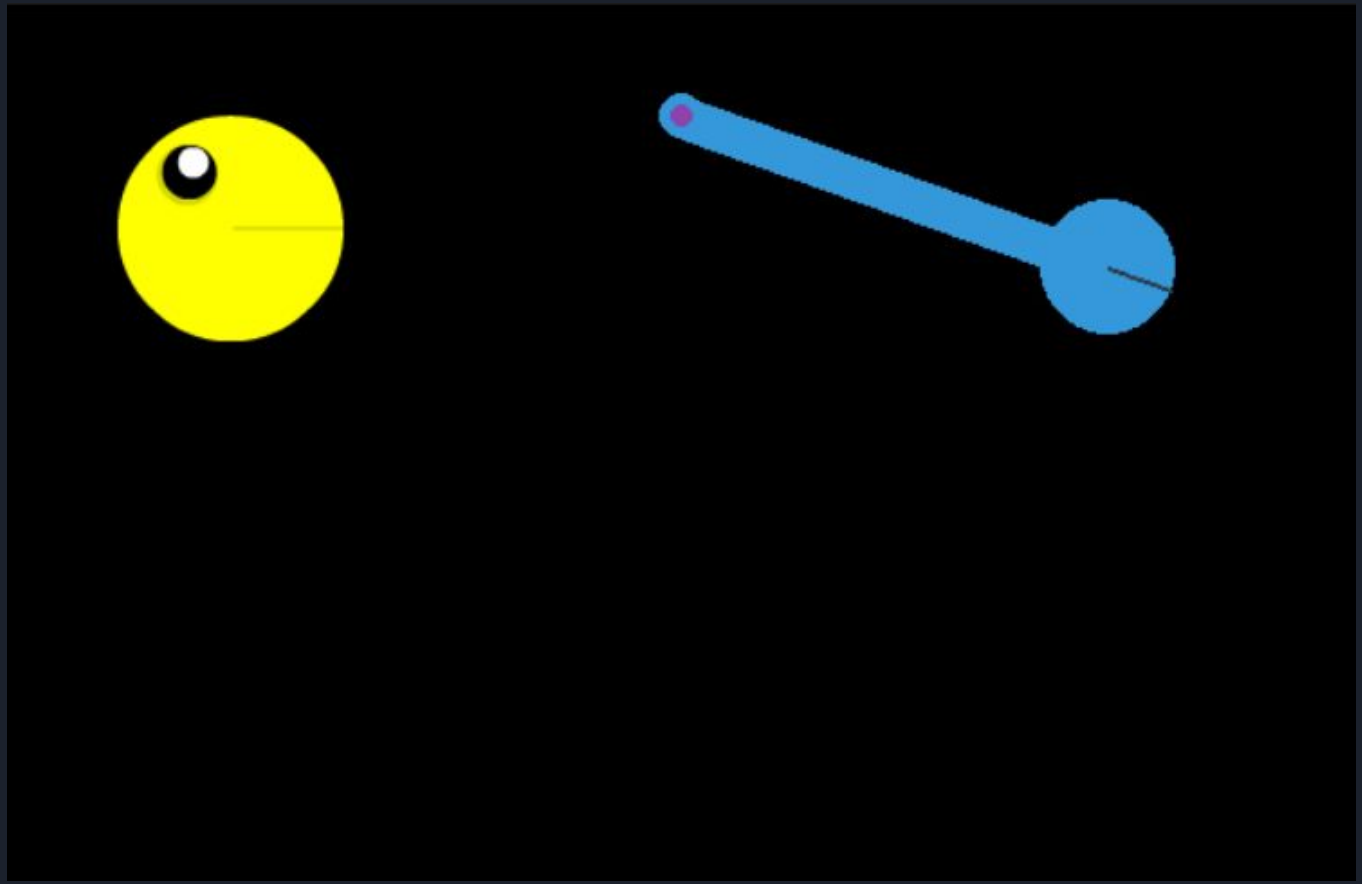


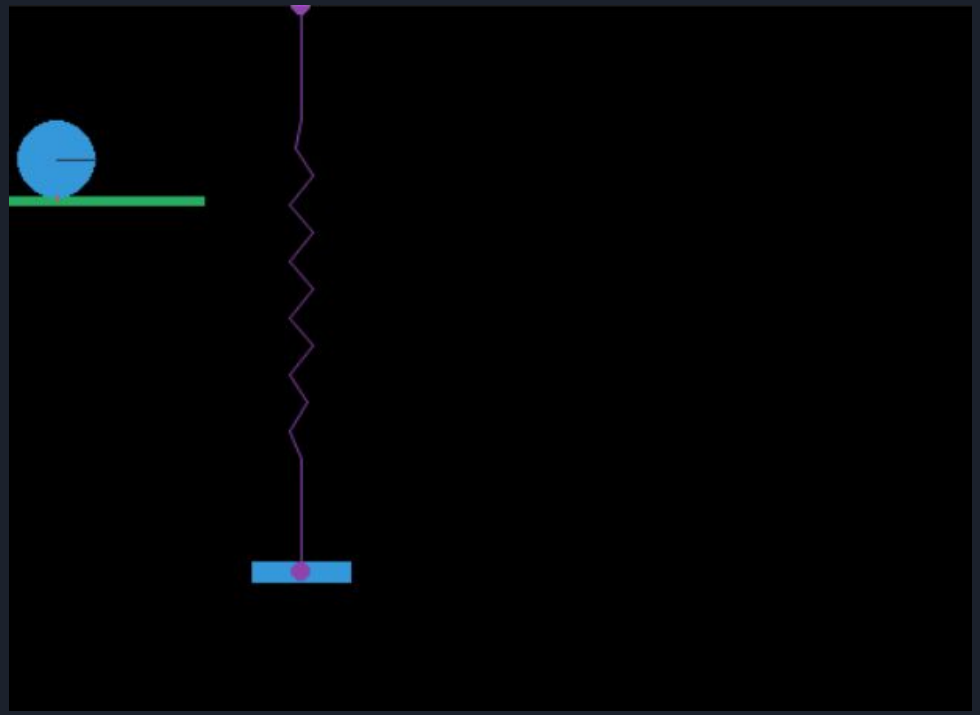
pkt	
1	0.18
10	0.07
100	0.21
5000	0.06
10000	0.05
20000	0.03
30000	0.10
40000	0.09
50000	0.13
100000	0.08


0.15
0.10
0.05
0.00
(-4.0, 1)
(-4.0, 10)
(-4.0, 100)
(-4.0, 5000)
(-4.0, 10000)
(-4.0, 20000)
(-4.0, 30000)
(-4.0, 40000)
(-4.0, 50000)
x_grav,pkt
0.3
0.2
0.1
0.0
(-3.0, 1)
(-3.0, 10)
(-3.0, 100)
(-3.0, 5000)
(-3.0, 10000)
(-3.0, 20000)
(-3.0, 30000)
(-3.0, 40000)
(-3.0, 50000)
x_grav,pkt
0.20
0.15
0.10
0.05
0.00
(-2.0, 1)
(-2.0, 10)
(-2.0, 100)
(-2.0, 5000)
(-2.0, 10000)
(-2.0, 20000)
(-2.0, 30000)
(-2.0, 40000)
(-2.0, 50000)
(-2.0, 100000)
x_grav,pkt
0.20
0.15
0.10
0.05
0.00
(-1.0, 1)
(-1.0, 10)
(-1.0, 100)
(-1.0, 5000)
(-1.0, 10000)
(-1.0, 20000)
(-1.0, 30000)
(-1.0, 40000)
(-1.0, 50000)
(-1.0, 100000)
x_grav,pkt
0.3
0.2
0.1
0.0
(1.0, 1)
(1.0, 10)
(1.0, 100)
(1.0, 5000)
(1.0, 10000)
(1.0, 20000)
(1.0, 30000)
(1.0, 40000)
(1.0, 50000)
(1.0, 100000)
x_grav,pkt
0.15
0.10
0.05
0.00
(2.0, 1)
(2.0, 10)
(2.0, 100)
(2.0, 5000)
(2.0, 10000)
(2.0, 20000)
(2.0, 30000)
(2.0, 40000)
(2.0, 50000)
(2.0, 100000)
x_grav,pkt
0.20
0.15
0.10
0.05
0.00
(3.0, 1)
(3.0, 10)
(3.0, 100)
(3.0, 5000)
(3.0, 10000)
(3.0, 20000)
(3.0, 30000)
(3.0, 40000)
(3.0, 50000)
(3.0, 100000)
x_grav,pkt
0.2
0.1
0.0
(4.0, 1)
(4.0, 10)
(4.0, 100)
(4.0, 5000)
(4.0, 10000)
(4.0, 20000)
(4.0, 30000)
(4.0, 40000)
(4.0, 50000)
(4.0, 100000)
x_grav,pkt

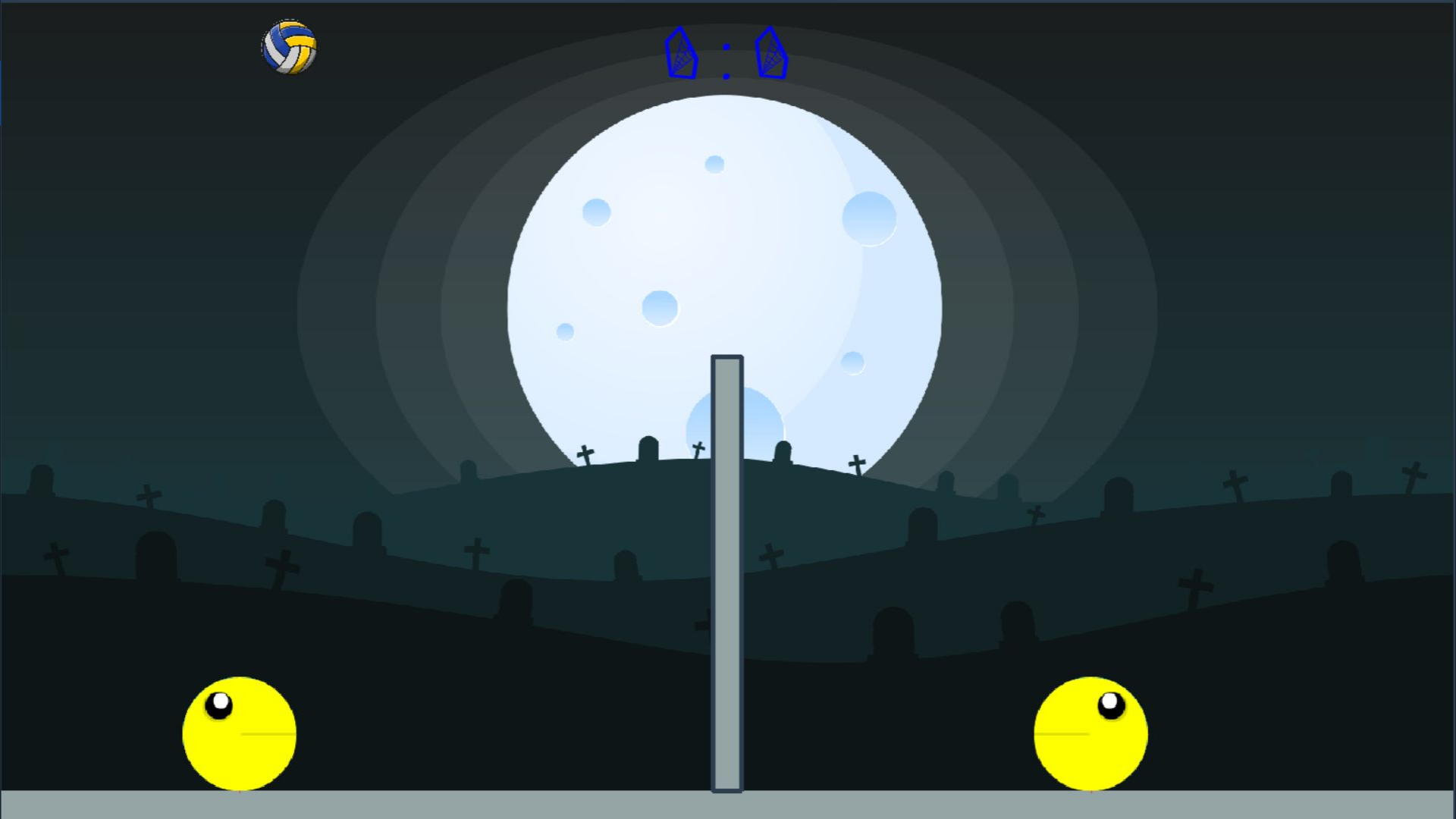
Przykłady







Przeciwnicy: 4



Dziękuję

Przygotował: Grzegorz Gołyś

e-mail: grzegorz97@opoczta.pl , grzegorz.golys@gmail.com

YouTube: Nauka z Gołym