Группа M3112 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К работе допущен\_\_\_\_\_\_\_

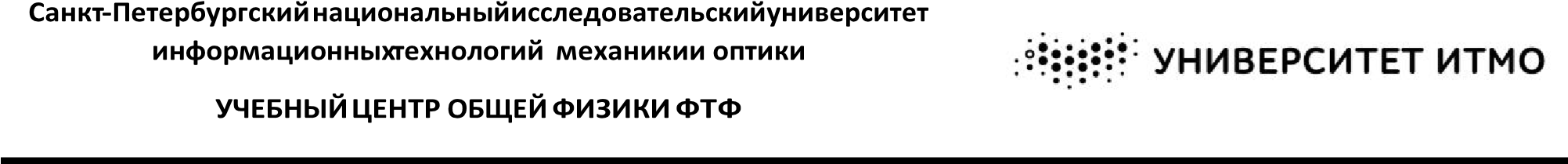
Студент Захаров Даниил Антонович & Работа выполнена \_\_\_\_\_\_\_

Преподаватель Шоев В.И.\_\_ Отчет принят \_

Рабочий протокол и отчет по моделированию на тему «Бросок тела с высоты, с учетом действия силы Кориолиса»

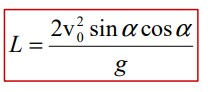
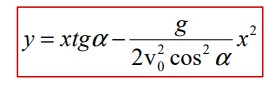
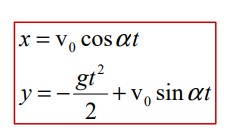
Использованные формулы: Сила Кориолиса:



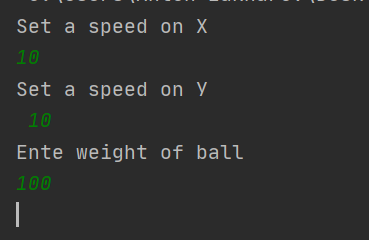
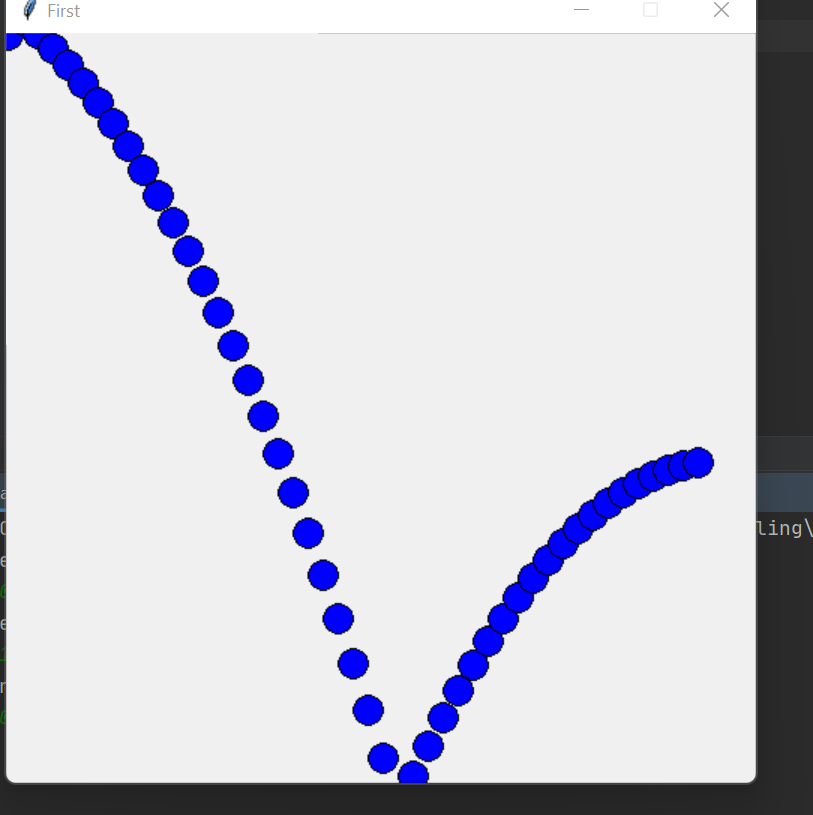
Кориолисово ускорение:



Формулы траектории для тела, брошенного с высоты:

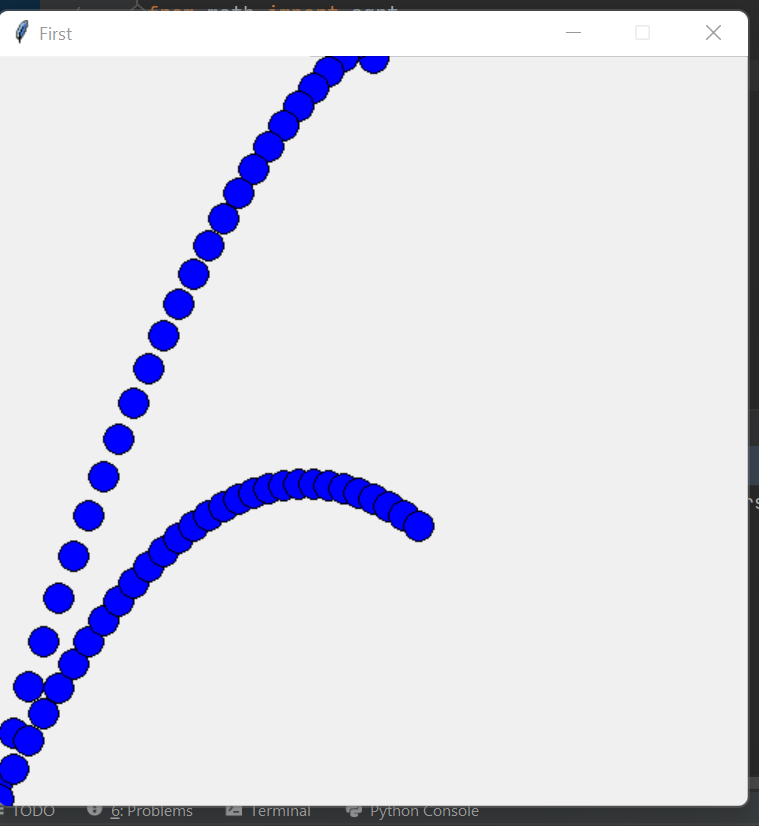


Результат работы программы при входных данных:

Все величины в эталонных единицах.

При том если мы введем отрицательную скорость по х, получим следующее:



Задача на ЗСИp = p1 + p2 (отскакивавания от стены)

ЗСЭ Е = Еп + Ек и преобразавнование(потеря энергии), часть энергии уходит на нагревание шара, в сследствии чего мяч при следующем прыжке подлетает на меньшую высоту

Если отключим потерю энергии, сделаем нашу систему «идеальной», мяч будет иметь не затухающие отскакивания

Что доказывает, что программа функционирует верно.

Ограничения на входные данные:

Масса > 0

Скорость по у =0

Скорость по х != 0

Листинг кода:

import graphics as g  
import time  
import keyboard  
from math import sqrt  
  
x, y = 250, 1  
speed\_x= input('Set a speed on X \n')  
speed\_y =input('Set a speed on У\n ')  
m = input('Ente weight of ball\n')  
# speed\_x, speed\_y = 0, 0  
  
  
SIZE\_X = 500  
SIZE\_Y = 500  
  
window = g.GraphWin("First", SIZE\_X, SIZE\_Y)  
  
coord = g.Point(x, y)  
velocity = g.Point(speed\_x, speed\_y)  
tyga = g.Point(0, 0.98)  
t = 0  
  
  
# координаты тела  
def add(point\_1, point\_2):  
 new\_point = g.Point(point\_1.x + point\_2.x,  
 round(point\_1.y - point\_2.y))  
  
 return new\_point  
  
  
# создание тела  
def draw\_ball(coord):  
 telo = g.Circle(coord, 10)  
 telo.setFill('blue')  
 telo.draw(window)  
  
  
# чистка окна после завершения  
def clear\_window():  
 rectangle = g.Rectangle(g.Point(0, 0), g.Point(SIZE\_X, SIZE\_Y))  
 rectangle.setFill('white')  
 rectangle.draw(window)  
  
  
# проверка координат для фбсолютно упргого сопротивления  
def check\_coord(coord, velocity, cos=228, sin=228):  
 if cos == 228 and sin == 228:  
 if coord.y < 0 or coord.y > SIZE\_Y:  
 velocity.y = - velocity.y  
 if coord.x < 0 or coord.x > SIZE\_X:  
 velocity.x = - velocity.x  
 if coord.y == SIZE\_Y:  
 velocity.y = - velocity.y // 2  
 else:  
 if coord.y < 0 or coord.y > SIZE\_Y:  
 velocity.y = -(velocity.y \* sin) - 9.8  
 if coord.x < 0 or coord.x > SIZE\_X:  
 velocity.x = -velocity.x  
 # clear\_window()  
 if coord.y == SIZE\_Y:  
 velocity.y = (-velocity.y // 2)  
  
  
# if coord.x == SIZE\_X or coord.x == 0:  
# clear\_window()  
  
  
# Tyga coord  
def coord2(coord, velocity):  
 return add(coord, velocity)  
  
  
# F тяг  
def velocity2(velocity, tyga):  
 return add(velocity, tyga)  
  
  
def find\_vector\_sin\_cos(last\_coords, current\_coords):  
 v\_x = g.Point(current\_coords.x - last\_coords.x, 0)  
 vector\_coords = g.Point(current\_coords.x - last\_coords.x, current\_coords.y - last\_coords.y)  
 cos = ((v\_x.x \* vector\_coords.x) + (v\_x.y \* vector\_coords.y)) / (  
 sqrt((v\_x.x \*\* 2) + (v\_x.y \*\* 2)) \* sqrt((vector\_coords.x \*\* 2) + (vector\_coords.y \*\* 2)))  
 sin = 1 - (cos \*\* 2)  
 return cos, sin  
  
  
# живность процессу(фпс) - еудщ проги  
while True:  
  
 # clear\_window()  
 draw\_ball(coord)  
  
 if t == 0:  
 last\_coordinates = coord  
 coord = coord2(coord, velocity)  
 velocity = velocity2(velocity, tyga)  
 check\_coord(coord, velocity)  
 else:  
 cos, sin = find\_vector\_sin\_cos(last\_coordinates, coord)  
 last\_coordinates = coord  
 coord = coord2(coord, velocity)  
 velocity = velocity2(velocity, tyga)  
 check\_coord(coord, velocity, cos=cos, sin=sin)  
  
 t += 1  
  
 g.time.sleep(0.1)