Выполнил Шеин Максим Андреевич М3106

Рабочий протокол и отчёт по моделированию №1

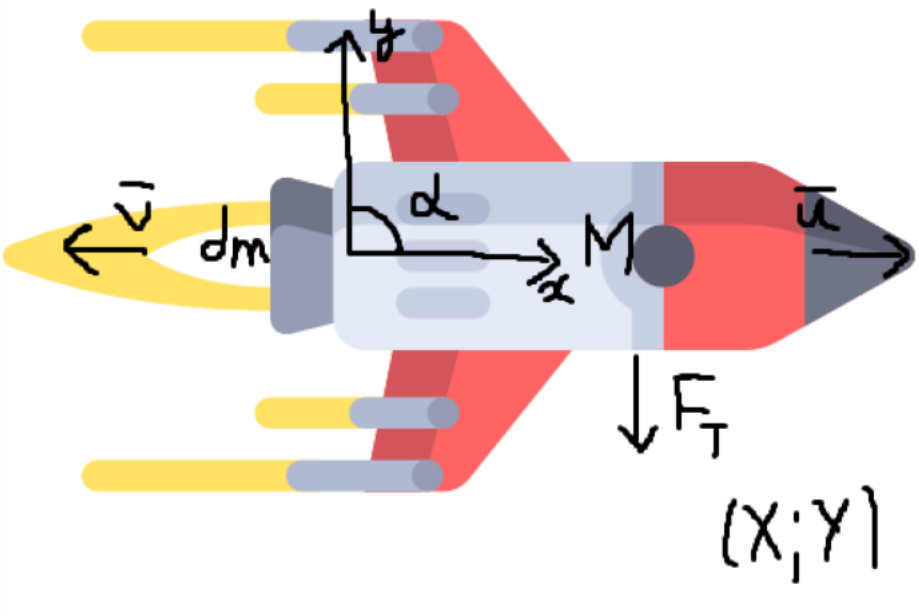
«Взлёт ракеты»

1)Задачи

Вывести координаты ракеты за каждый тик
Скорость ракеты по оси X и оси Y на каждый тик
Массу топлива на каждый тик
Высчитать ускорение свободного падения для данной планеты

Вначале мы считаем именение скоростей, затем мы высчитываем новый угол нашей ракеты, и в конце считаем новые координаты

2)Используемые переменные



v_x, v_y – изменение скорости ракеты по x и y

M – масса ракеты

dm – масса топлива, которая вылетела

α – угол наклона ракеты

X, Y – координаты ракеты

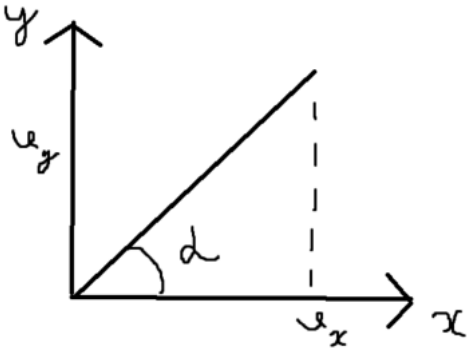
3)Формулы

3.1)Изменение скоростей

$$dm \cdot \bar{v} = (M - dm) \cdot \bar{u} \qquad u_x = \frac{dm \cdot v_x}{M - dm}$$

$$\bar{u} = \frac{dm \cdot \bar{v}}{M - dm} \qquad u_y = \frac{dm \cdot v_y}{M - dm}$$

3.2)Подсчёт нового угла



$$\alpha = \arctan \frac{u_y}{u_x}$$

3.3)Расчёт координат

$$X+ = u_x$$

$$Y+ = u_y$$

3.4)Гравитационная сила

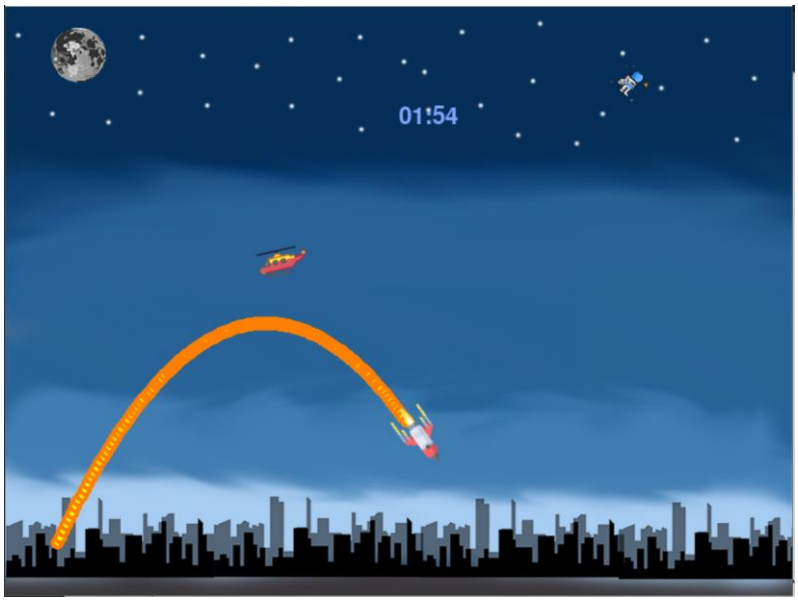
$$g = G \cdot \frac{M}{R^2}$$

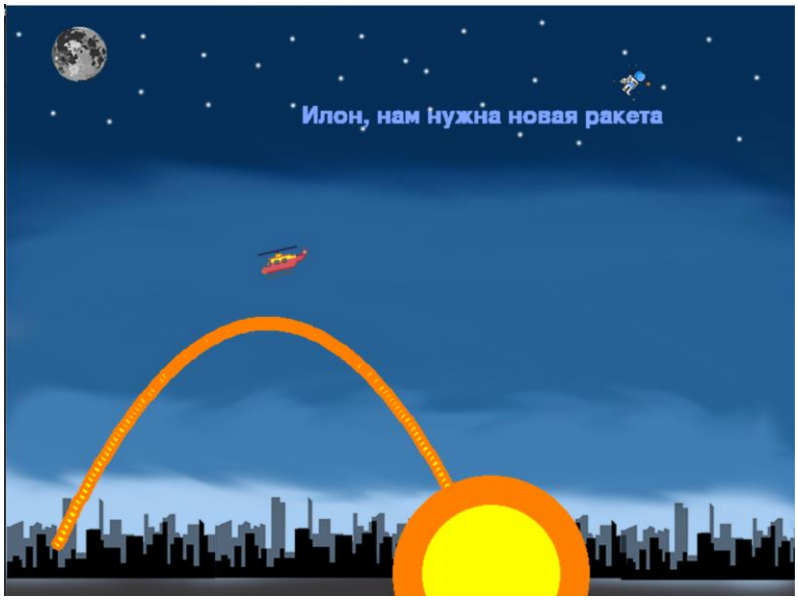
G = 6,6720 – гравитационная постоянная

M – масса планеты

R – радиус планеты

4)Схема установки





5)Расчёт результатов

```
00:07
g= 9.475911335762666
X = 23772.019377002158
Y = 128807.42108543881
Скорость по X = 583.3019655158433
Скорость по Y = 6590.775030566396
Масса топлива = 0
Угол = 84.94234141523245
```

6)Код

```
import pygame as pg
import math
import tkinter as tk

MassOfThePlanet = #Ваше число - Масса планеты
RadiusOfThePlanet = 6371000 # Радиус планеты
g = 6.6720 * MassOfThePlanet / (RadiusOfThePlanet ** 2) / 10 ** 11 #
Гравитационная сила

# ------------------------------------------------------------------------
----------------------------------------
# Мини окно для переназначения начальных значений
config = [x, y, z, a, b, c] # Изначальные значения

def clicked():
    global config
    for ind in range(len(config)):
        config[ind] = int(entry_list[ind].get())
    window_configuration.destroy()


window_configuration = tk.Tk()
window_configuration.title("MaksSheinM3106")
window_configuration.geometry('250x160')

tk.Label(window_configuration, text="Стартовый угол").grid(column=0, row=0)
tk.Label(window_configuration, text="Масса ракеты").grid(column=0, row=1)
tk.Label(window_configuration, text="Масса топлива").grid(column=0, row=2)
tk.Label(window_configuration, text="Скорость истечения газа").grid(column=0,
row=3)
tk.Label(window_configuration, text="Сторость сжигания
топлива").grid(column=0, row=4)
tk.Label(window_configuration, text="Масштаб").grid(column=0, row=5)


entry_list = []
for i in range(6):
    entry_list.append(tk.Entry(window_configuration, width=10))

for i in range(6):
    entry_list[i].grid(column=1, row=i)

tk.Button(window_configuration, text="ПУСК!", command=clicked).grid(column=0,
row=12)
window_configuration.mainloop()
```

```
pg.font.init()
pg.init()
#-------------------------------------------------------------------------------
-----------------------------------------
#Базовые значения
fps = 60 # тут итак всё понятно
width = 800 # Ширина окна
height = 600 # Высота окна
FontSize = 36 # Размер шрифта
FontColor = (128, 166, 255) # Цвет шрифта
# ------------------------------------------------------------------------------
-----------------------------------------
# Загрузка модельки ракеты и заднего фона
rocket_img = pg.image.load("rocket.png")
font = pg.font.Font(None, FontSize)
bg = pg.image.load("back.png")
scale = config[5]
# ------------------------------------------------------------------------------
-----------------------------------------
# Ракетный след
class Point:
    def __init__(self, x, y):
        self.x = x // scale
        self.y = y // scale

    def trail(self, scr, biasX_, biasY_):
        pg.draw.circle(scr, (255, 128, 0), (int(self.x + biasX_), int(height
- self.y + biasY_)), 7)
        pg.draw.circle(scr, (255, 255, 0), (int(self.x + biasX_), int(height
- self.y + biasY_)), 3)

    def Boom(self, scr, biasX_, biasY_):
        pg.draw.circle(scr, (255, 128, 0), (int(self.x + biasX_), int(height
- self.y + biasY_)), 100)
        pg.draw.circle(scr, (255, 255, 0), (int(self.x + biasX_), int(height
- self.y + biasY_)), 70)

# Ракета
class Rocket:
    def __init__(self, angle, rocketW, fuelW, GasVelocity, CombustionRate,
image, width, height):
        self.angle = angle
        self.DifferentAngle = angle
        self.rocketW = rocketW + fuelW
        self.fuelW = fuelW
        self.GasVelocity = GasVelocity # Скорость истечения газа из сопла
ракеты
        self.CombustionRate = CombustionRate # Скорость сгорания
        self.image = image
        self.width = width
        self.height = height
        self.x = 20 * scale
        self.y = (20 + self.height) * scale
        self.velocityX = 0
        self.velocityY = 0

    # Гравитационная сила
    def changeGravity(self):
        global MassOfThePlanet, RadiusOfThePlanet, g
        g = 6.6720 * MassOfThePlanet / ((RadiusOfThePlanet + self.y) ** 2) /
10 ** 11

    # Гравитация
    def gravity(self):
```

```
        global fps, g
        self.velocityY -= g / fps

    # Изменение угла
    def changeAngel(self):
        self.DifferentAngle = math.atan2(self.velocityY, self.velocityX) *
180 / math.pi

    # Изменение скорости
    def changeVelocity(self):
        global fps
        self.velocityX += (self.CombustionRate / fps) * (self.GasVelocity *
math.cos(self.angle * math.pi / 180)) / (
                self.rocketW - (self.CombustionRate / fps))
        self.velocityY += (self.CombustionRate / fps) * (self.GasVelocity *
math.sin(self.angle * math.pi / 180)) / (
                self.rocketW - (self.CombustionRate / fps))

    # Изменение координат
    def changeCoordinates(self):
        global fps
        self.x += self.velocityX / fps
        self.y += self.velocityY / fps

    # Изменение расположения ракетного следа
    def blit(self, scr):
        rotate_img = pg.transform.rotate(self.image, -90 +
self.DifferentAngle)
        rect = self.image.get_rect(topleft=(int(self.x // scale), int(height
- (self.y // scale))))
        scr.blit(rotate_img, rect)

    # Состояние ракеты
    def checkCollision(self):
        if (self.y // scale) - self.height // 2 < 20:
            return "BOOOM"
        if (self.y // scale) - self.height // 2 > height:
            return "SUCCESS"
        return "FLIGHT"

# ------------------------------------------------------------------------
----------------------------------------
# Функция, отвечающая за состояние ракеты
def pause(message):
    global run
    isPause = True
    while isPause:
        for ev in pg.event.get():
            if ev.type == pg.QUIT:
                isPause = False
                run = False
            elif ev.type == pg.KEYDOWN:
                if ev.key == pg.K_SPACE:
                    isPause = False
                    run = False
        text = font.render(message, True, FontColor)
        screen.blit(text, (300, 100))
        pg.display.update()
# ------------------------------------------------------------------------
----------------------------------------

screen = pg.display.set_mode((width, height))
clock = pg.time.Clock()
```

```
run = 1 # Изначально, запуск ракеты - тру

rocket = Rocket(config[0], config[1], config[2], config[3], config[4],
rocket_img, rocket_img.get_width(), rocket_img.get_height())
trajectory = []
# Таймер
t = 0
second = 0
minute = 0
sec_str = "00"
min_str = "00"
timer = ""
s = font.render("00:00", True, FontColor)

while run:
    clock.tick(fps) # тики
    rocket.changeGravity() # Гравитационаня сила
    screen.fill((0, 0, 0)) # фон(на нуле, т.к. у нас уже он есть)
    screen.blit(bg, (0, 0))
    for event in pg.event.get():
        if event.type == pg.QUIT: # Если закрыть окно, программа пректит
процесс выполнения
            run = False
        elif event.type == pg.KEYDOWN:
            if event.key == pg.K_SPACE: # Если нажать пробел, программа
пректит процесс выполнения
                run = False

    # Если топливо ещё есть
    if rocket.fuelW > 0:
        rocket.fuelW -= rocket.CombustionRate / fps
        rocket.changeVelocity()

    # Если ракета летит
    if rocket.y > 30 + rocket.height:
        rocket.gravity()
    rocket.changeAngel()
    rocket.changeCoordinates()
    t = (t + 1) % fps

    #Информация
    if t == 0:
        second += 1
        trajectory.append(Point(rocket.x, rocket.y))
        print("g=", g
              , "\nX =", rocket.x
              , "\nY =", rocket.y
              , "\nСкорость по X =", rocket.velocityX
              , "\nСкорость по Y =", rocket.velocityY
              , "\nМасса топлива =", max(rocket.fuelW, 0)
              , "\nУгол =", rocket.DifferentAngle)

        if second % 60 == 0:
            minute += 1
            second = 0
            if minute < 10:
                min_str = "0" + str(minute)
            else:
                min_str = str(minute)

        if second < 10:
            sec_str = "0" + str(second)
        else:
            sec_str = str(second)
```

```
        timer = min_str + ":" + sec_str
        print("\n" + timer)
        s = font.render(timer, True, FontColor)

    for point in trajectory:
        point.trail(screen, rocket.width // 2, rocket.height // 2)
    rocket.blit(screen)

    state = rocket.checkCollision()
    if state == "BOOOM":
        point.Boom(screen, rocket.width // 2, rocket.height // 2)
        pause("Илон, нам нужна новая ракета")
    if state == "SUCCESS":
        pause("Ракета в космосе!")
    screen.blit(s, (400, 100))
    pg.display.update()
```