|  |  |  |
| --- | --- | --- |
| Защищено:  Гапанюк Ю. Е.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |  | Демонстрация:  Гапанюк Ю. Е.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

**Отчет по лабораторной работе №4 по курсу**

**«Парадигмы и конструкции языков программирования»**

#### Тема работы: " **Работа с анимацией** "

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: | Пыжьянов Александр Сергеевич |
| студент группы  ИУ5Ц-52Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Гапанюк Ю.Е. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

Москва, МГТУ 2024

**СОДЕРЖАНИЕ**

[Цель лабораторной работы 3](#_Toc176887611)

[Практическое задание 3](#_Toc176887612)

[Листинг программы 4](#_Toc176887613)

[Результат работы программы 15](#_Toc176887614)

## Цель лабораторной работы

Научиться работе с библиотекой Pygame.

## Практическое задание

Поиграться с анимашкой.

## Листинг программы

import pygame

import math

pygame.init()

width, height = 800, 600

screen = pygame.display.set\_mode((width, height))

pygame.display.set\_caption('Корабль по синусоиде')

WHITE = (255, 255, 255)

BLUE = (0, 0, 255)

RED = (255, 0, 0)

BLACK = (0, 0, 0)

GREEN = (0, 255, 0)

amplitude = 100

frequency = 0.01

speed = 10

ship\_width = 60

ship\_height = 30

flag\_height = 20

x = 0

y = height // 2

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

x += speed

y = height // 2 + amplitude \* math.sin(frequency \* x)

if x > width:

x = -ship\_width

screen.fill(WHITE)

for i in range(width):

sine\_y = height // 2 + amplitude \* math.sin(frequency \* i)

pygame.draw.circle(screen, GREEN, (i, int(sine\_y)), 1)

pygame.draw.ellipse(screen, BLUE, (x, y, ship\_width, ship\_height))

pygame.draw.rect(screen, RED, (x+ship\_width // 2 - 2, y - flag\_height, 4, flag\_height))

pygame.draw.polygon(screen, RED, [

(x + ship\_width // 2 - 2, y - flag\_height),

(x + ship\_width // 2 - 2, y - flag\_height + 10),

(x + ship\_width // 2 + 10, y - flag\_height + 5)

])

pygame.display.flip()

pygame.time.Clock().tick(60)

pygame.quit()

import pygame

import math

pygame.init()

w, h = 800, 600

screen = pygame.display.set\_mode((w, h))

pygame.display.set\_caption("Квадрат движется по полукругу и вращается")

BLACK = (0, 0, 0)

WHITE = (255, 255, 255)

sz = 50

c\_x, c\_y = w // 2, h // 2

rad = 200

ang\_sp = 0.02

rot\_sp = 5

angle = 0

rotation\_angle = 0

direction = 1

running = True

while running:

for event in pygame.event.get():

if event.type == pygame.QUIT:

running = False

screen.fill(BLACK)

x = c\_x + rad \* math.cos(angle) - sz // 2

y = c\_y + rad \* math.sin(angle) - sz // 2

square = pygame.Surface((sz, sz), pygame.SRCALPHA)

square.fill(WHITE)

rotated\_square = pygame.transform.rotate(square, rotation\_angle)

rect = rotated\_square.get\_rect(center=(x + sz // 2, y + sz // 2))

screen.blit(rotated\_square, rect.topleft)

angle += ang\_sp \* direction

rotation\_angle += rot\_sp

if angle > math.pi or angle < 0:

direction \*= -1

angle += ang\_sp \* direction

pygame.display.flip()

pygame.time.Clock().tick(60)

pygame.quit()

import pygame

import sys

import math

pygame.init()

# Размеры окна

WIDTH, HEIGHT = 540, 960

# Количество кадров в секунду

FPS = 60

# Создание окна

screen = pygame.display.set\_mode((WIDTH, HEIGHT))

pygame.display.set\_caption('Анимация')

# Загрузка изображений

background\_img = pygame.image.load('background.png')

starship\_img = pygame.image.load('starship.png')

fire\_img = pygame.image.load('fire.png')

# Получение размеров фонового изображения

bg\_width = background\_img.get\_width()

bg\_height = background\_img.get\_height()

# Начальные координаты фоновых изображений

bg\_y1 = 0

bg\_y2 = -bg\_height

# Скорость прокрутки фона

bg\_speed = 3

# Положение звездолета

starship\_rect = starship\_img.get\_rect()

starship\_rect.centerx = WIDTH / 2 # Центрирование по горизонтали

clock = pygame.time.Clock()

time = 0 # Переменная времени для анимации

while True:

for event in pygame.event.get():

if event.type == pygame.QUIT:

pygame.quit()

sys.exit()

# Обновляем переменную времени

time += 1 / FPS # Увеличиваем время на долю секунды, равную одному кадру (1/FPS)

# Вычисляем синусоиду текущего времени

sine\_offset = math.sin(time) # Значение синуса времени колеблется между -1 и 1

sine\_offset\_inverse = 1 - sine\_offset # Инвертированное значение синусоиды для противоположного эффекта

# Двигаем звездолет по синусоиде вверх-вниз

starship\_rect.y = HEIGHT / 2 + 100 \* sine\_offset # Центрируем звездолет и добавляем синусоиду для вертикального движения

# Обновляем позиции фона для бесконечной прокрутки, изменяя скорость на основе синусоиды

bg\_y1 += bg\_speed + 4 \* sine\_offset\_inverse # Увеличиваем скорость прокрутки фона в зависимости от синусоиды

bg\_y2 += bg\_speed + 4 \* sine\_offset\_inverse

if bg\_y1 >= HEIGHT:

bg\_y1 = bg\_y2 - bg\_height # Перемещаем фон наверх, если он вышел за нижнюю границу экрана

if bg\_y2 >= HEIGHT:

bg\_y2 = bg\_y1 - bg\_height # Перемещаем фон наверх, если он вышел за нижнюю границу экрана

# Позиция анимации огня под звездолетом

fire\_pos = (starship\_rect.centerx - fire\_img.get\_width() / 2,

starship\_rect.centery + 10 \* sine\_offset\_inverse) # Положение огня немного смещается по синусоиде

# Отрисовываем все элементы на экране

screen.blit(background\_img, (0, bg\_y1))

screen.blit(background\_img, (0, bg\_y2))

screen.blit(fire\_img, fire\_pos)

screen.blit(starship\_img, starship\_rect.topleft)

pygame.display.flip()

clock.tick(FPS) # Ограничиваем обновление экрана до 60 кадров в секунду

import tkinter as tk

from tkinter import ttk

def visual():

def save\_r1():

entry\_r1.config(state = 'readonly')

def clear\_r1():

entry\_r1.config(state = 'normal')

entry\_r1.delete('0', 'end')

def save\_r2():

entry\_r2.config(state = 'readonly')

def clear\_r2():

entry\_r2.config(state = 'normal')

entry\_r2.delete('0', 'end')

def save\_tip():

vibor\_soed.config(state = 'disabled')

def clear\_tip():

vibor\_soed.config(state = 'readonly')

vibor\_soed.delete('0', 'end')

def clear\_ans():

label\_ans.config(text = '')

def podschet():

r1 = entry\_r1.get()

r2 = entry\_r2.get()

tip = vibor\_soed.get()

try:

r1 = int(r1)

r2 = int(r2)

if tip == 'Последовательное':

ans = r1 + r2

label\_ans.config(text = str(ans))

else:

ans = 1 / (1 / r1 + 1 / r2)

label\_ans.config(text = str(ans))

except:

label\_ans.config(text = 'Некорректный ввод')

def end\_prog():

root.destroy()

root = tk.Tk() #создание окна

root.title('Подсчет сопротивления с двух резисторов') #заголовок

root.geometry('900x600') #исходный размер

label\_vvod\_1 = tk.Label(text = 'Введите сопротивление первого резистора')

entry\_r1 = tk.Entry() #создание поля для ввода

btn\_vvod\_1 = ttk.Button(text = 'Ввод', command = save\_r1) #создание кнопки

btn\_vvod\_clear\_r1 = ttk.Button(text = 'Очистить', command = clear\_r1)

label\_vvod\_1.grid(row = 0, column = 0, sticky = 'w')

entry\_r1.grid(row = 0, column = 1, sticky = 'w')

btn\_vvod\_1.grid(row = 0, column = 2, sticky = 'w')

btn\_vvod\_clear\_r1.grid(row = 0, column = 3, sticky = 'w')

label\_vvod\_2 = tk.Label(text = 'Введите сопротивление второго резистора')

entry\_r2 = tk.Entry() #создание поля для ввода

btn\_vvod\_2 = ttk.Button(text = 'Ввод', command = save\_r2) #создание кнопки

btn\_vvod\_clear\_r2 = ttk.Button(text = 'Очистить', command = clear\_r2)

label\_vvod\_2.grid(row = 1, column = 0, sticky = 'w')

entry\_r2.grid(row = 1, column = 1, sticky = 'w')

btn\_vvod\_2.grid(row = 1, column = 2, sticky = 'w')

btn\_vvod\_clear\_r2.grid(row = 1, column = 3, sticky = 'w')

label\_soed = tk.Label(text = 'Выберите тип соединения')

tip = ('Последовательное', 'Параллельное')

vibor\_soed = ttk.Combobox(values = tip, state = 'readonly') #создание мультивыбора

btn\_vibor = ttk.Button(text = 'Ввод', command = save\_tip)

vibor\_soed\_clear = ttk.Button(text = 'Очистить', command = clear\_tip)

label\_soed.grid(row = 2, column = 0, sticky = 'w')

vibor\_soed.grid(row = 2, column = 1, sticky = 'w')

btn\_vibor.grid(row = 2, column = 2, sticky = 'w')

vibor\_soed\_clear.grid(row = 2, column = 3, sticky = 'w')

label\_itog = tk.Label(text = 'Результат вычислений')

label\_ans = tk.Label(text = 'Не было вычислений')

btn\_ans = ttk.Button(text = 'Очистить', command = clear\_ans)

label\_itog.grid(row = 3, column = 0, sticky = 'w')

label\_ans.grid(row = 3, column = 1, sticky = 'w')

btn\_ans.grid(row = 3, column = 2, sticky = 'w')

btn\_deyst = ttk.Button(text = 'Выполнить', command = podschet)

btn\_deyst.grid(row = 4, column = 0, sticky = 'w')

btn\_end = ttk.Button(text = 'Завершить выполнение программы', command = end\_prog)

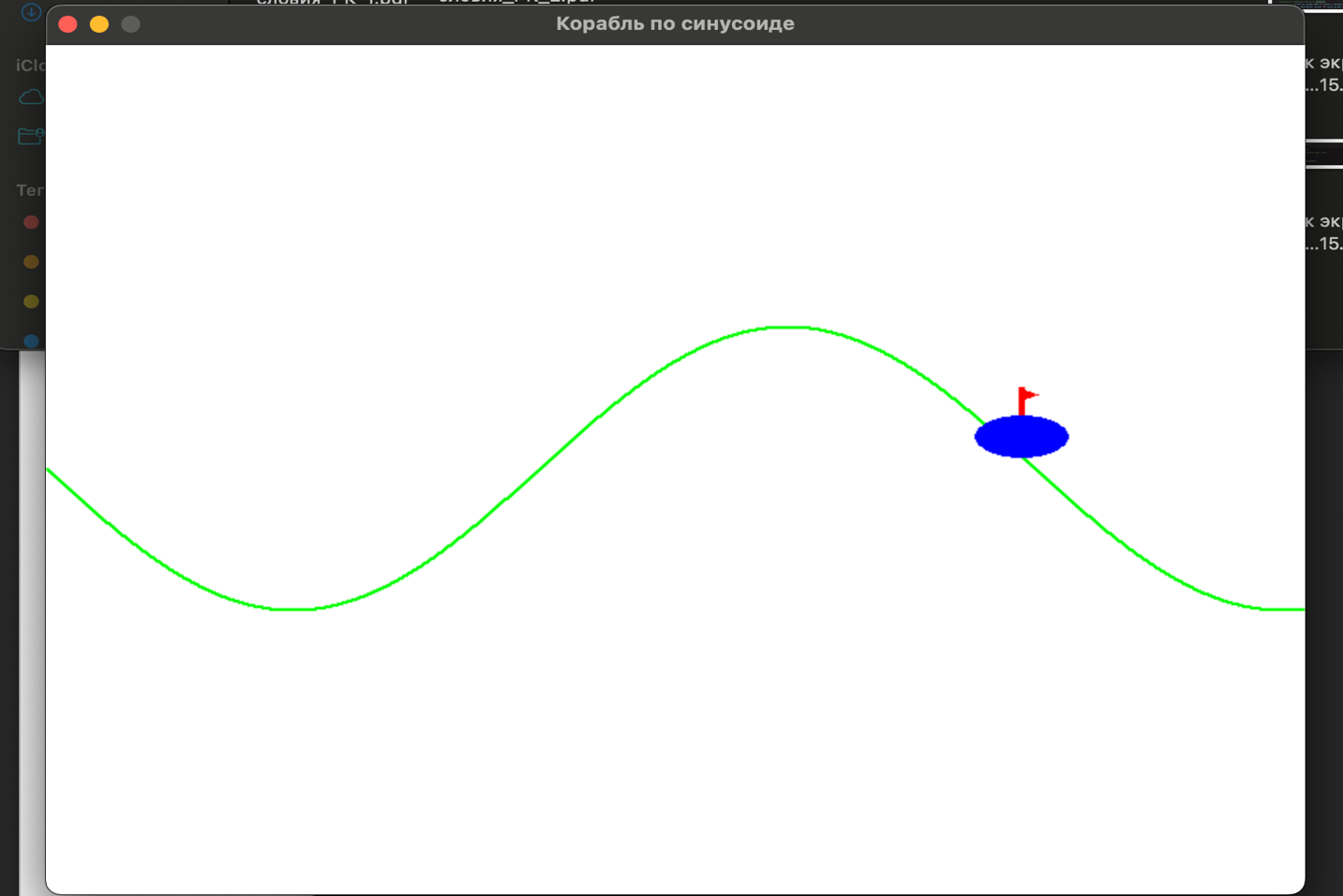
btn\_end.grid(row = 3, column = 4, sticky = 'w')

root.mainloop()

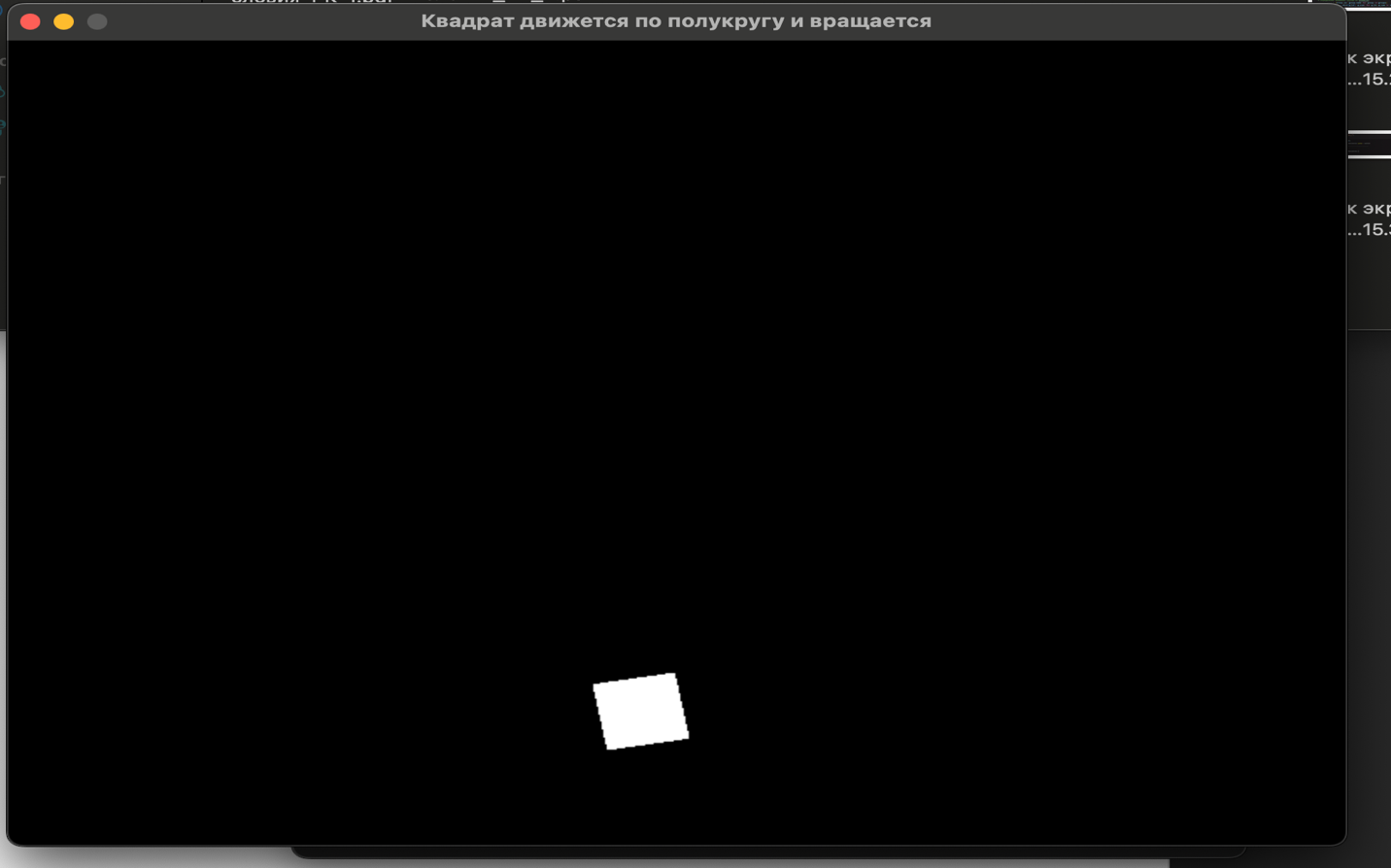
visual()

## Результат работы программы

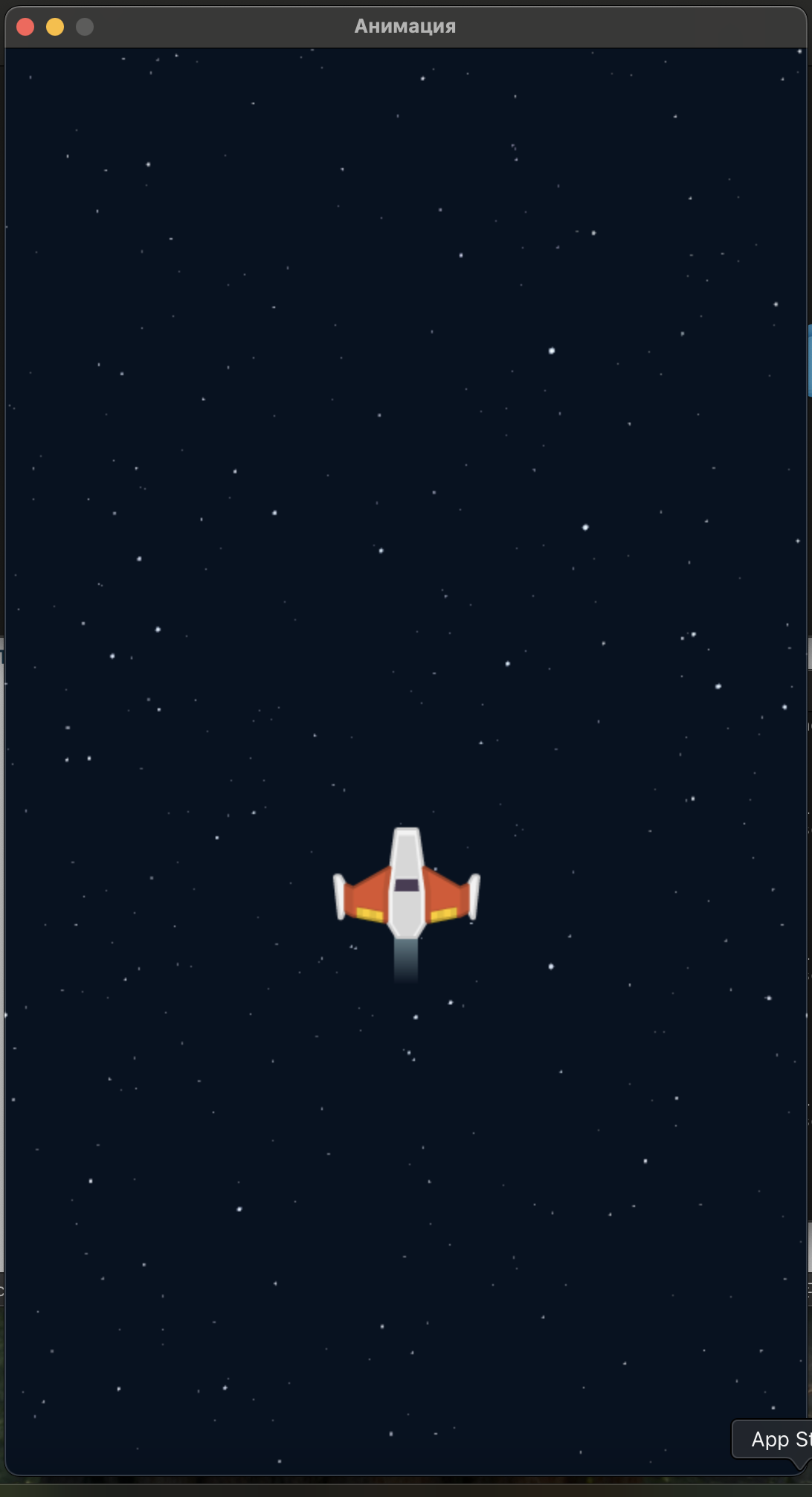
Для 1 кода



Для 2 кода



Для 3 кода



Для 4 кода

