Лабораторная работа № 3

Голик Елизавета Александровна
Номер студенческого билета : 242054
Название предмета : Практикум по программированию
Группа : ИД24-1
Дата сдачи : 25.11.2025
Название задачи : Превращение круга

B. Описание задания

С помощью полярных координат реализуйте следующее преобразование круга в треугольник.

C. Достигнутая сложность и реализация

Основные требования:

- Генерация точек по окружности.
- Формирование целевых точек — равностороннего треугольника.
- Плавная анимация перехода из круга в треугольник.
- Отрисовка соединённых линиями точек.
- Использование ООП
- Управление настройками через config.json

Дополнительная функциональность:

- Изменение цвета анимируемой фигуры через config.json.
- Изменяемая скорость анимации.
- Отображение текущего значения шага анимации (t) на экране.

Оригинальные расширения:

- Реализована корректная генерация треугольника равномерным распределением точек по периметру.
- Реализовано плавное движение через линейную интерполяцию.

D. Объяснение проектирования программы

1. Структура классов (ООП):

Класс Point:

- хранение координат
- преобразование из полярных координат
- плавное смещение move()

Класс CircleToTriangle:

- загрузка настроек
- генерация точек круга
- генерация точек треугольника
- обновление состояния анимации
- отрисовка

2. Организация Кода

Код разбит логически:

- загрузка конфигурации
- класс точки
- класс основного приложения
- метод run - это главный цикл
- вынесение параметров в config.json

3. Математическое исследование

1) Генерация круга

$$x = cx + r * \cos(angle)$$

$$y = cy + r * \sin(angle)$$

2) Генерация треугольника

Строятся три вершины равностороннего треугольника:

- A — верхняя вершина
- B и C — нижние углы

Точки распределяются равномерно вдоль сторон методом линейной интерполяции:

$$x = x1 + (x2 - x1) * t$$

$$y = y1 + (y2 - y1) * t$$

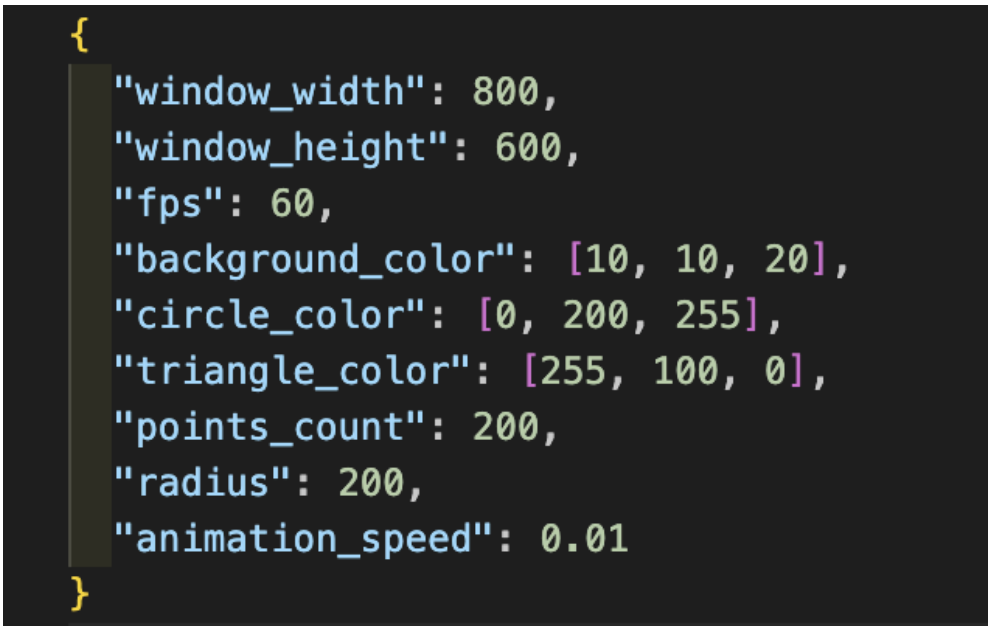
3) Плавное движение

Каждая точка движется к цели по формуле:

x += (target_x - x) * speed

y += (target_y - y) * speed

4. Управление настройками



```
{
    "window_width": 800,
    "window_height": 600,
    "fps": 60,
    "background_color": [10, 10, 20],
    "circle_color": [0, 200, 255],
    "triangle_color": [255, 100, 0],
    "points_count": 200,
    "radius": 200,
    "animation_speed": 0.01
}
```

Е. Основная часть (структурированная по требованиям)

Требование №1. Генерация точек круга

Решение:

Используются полярные координаты.

```
self.circle_points = [
    Point(angle=i * (2 * math.pi / self.points_count),
          radius=self.radius,
          center=self.center)
    for i in range(self.points_count)
]
```

Требование №2. Генерация треугольника

Решение:

Созданы три вершины и точки равномерно распределены по периметру.

```
def generate_triangle_points(self):
    A = (self.center[0], self.center[1] - self.radius)
    B = (self.center[0] - self.radius * math.sin(math.pi / 3),
        self.center[1] + self.radius * math.cos(math.pi / 3))
    C = (self.center[0] + self.radius * math.sin(math.pi / 3),
        self.center[1] + self.radius * math.cos(math.pi / 3))
```

Требование №3. Анимация преобразования

Решение:

Используется плавная интерполяция (linear interpolation).

```
p.move(tx, ty, self.speed)
```

Дополнение №1. “Изменение цвета”

Решение:

Цвет берётся из конфигурации.

```
self.triangle_color = config["triangle_color"]
```

Дополнение №2. “Изменяемая скорость”

```
self.speed = config["animation_speed"]
```

Дополнение №3. “Вывод шага анимации”

```
if self.show_step:
    text = self.font.render(f"t = {self.t:.2f}", True, (255, 255, 255))
    self.screen.blit(text, (10, 10))
```

F. Полный исходный код

```
import pygame

import json

import math

import sys



def load_config(path="config.json"):

   try:

       with open(path, "r") as f:

           return json.load(f)

   except Exception as e:

       print("Ошибка загрузки config.json:", e)

       sys.exit()



class Point:

   def __init__(self, angle, radius, center):

       self.angle = angle

       self.radius = radius

       self.center = center

       self.x, self.y = self.polar_to_cartesian(radius)



   def polar_to_cartesian(self, r):

       cx, cy = self.center

       return cx + r * math.cos(self.angle), cy + r * math.sin(self.angle)
```

```python
    def move(self, target_x, target_y, t):
        """Плавное движение к цели с коэффициентом t"""
        self.x += (target_x - self.x) * t
        self.y += (target_y - self.y) * t


# ГЛАВНЫЙ КЛАСС АНИМАЦИИ
class CircleToTriangle:
    def __init__(self, config):
        pygame.init()
        self.w = config["window_width"]
        self.h = config["window_height"]
        self.screen = pygame.display.set_mode((self.w, self.h))
        pygame.display.set_caption("Circle → Triangle")
        self.clock = pygame.time.Clock()
        self.fps = config["fps"]
        self.bg = config["background_color"]
        self.circle_color = config["circle_color"]
        self.triangle_color = config["triangle_color"]
        self.center = (self.w // 2, self.h // 2)
        self.points_count = config["points_count"]
        self.radius = config["radius"]
        self.speed = config["animation_speed"]
        self.t = 0  # Параметр перехода 0→1

        # Генерация точек на круге
```

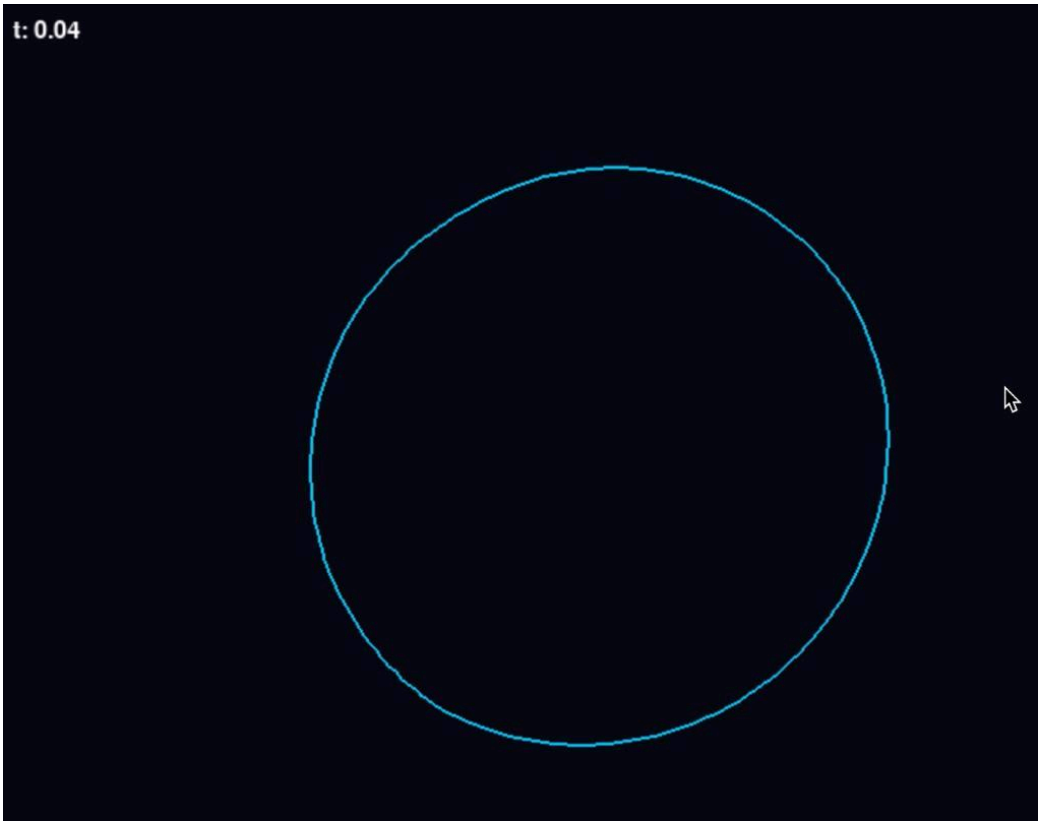
```
        self.circle_points = [
            Point(angle=i * (2 * math.pi / self.points_count),
                  radius=self.radius,
                  center=self.center)
            for i in range(self.points_count)
        ]

        # Целевые точки (треугольник)
        self.triangle_points = self.generate_triangle_points()

        # Шрифт для отображения t
        self.font = pygame.font.SysFont(None, 24)

    # Генерация точек треугольника
    def generate_triangle_points(self):
        """Создаём равносторонний треугольник по периметру"""
        A = (self.center[0], self.center[1] - self.radius)
        B = (self.center[0] - self.radius * math.sin(math.pi / 3),
             self.center[1] + self.radius * math.cos(math.pi / 3))
        C = (self.center[0] + self.radius * math.sin(math.pi / 3),
             self.center[1] + self.radius * math.cos(math.pi / 3))

        edges = [A, B, C]
        points = []

        segment_length = self.points_count // 3
```

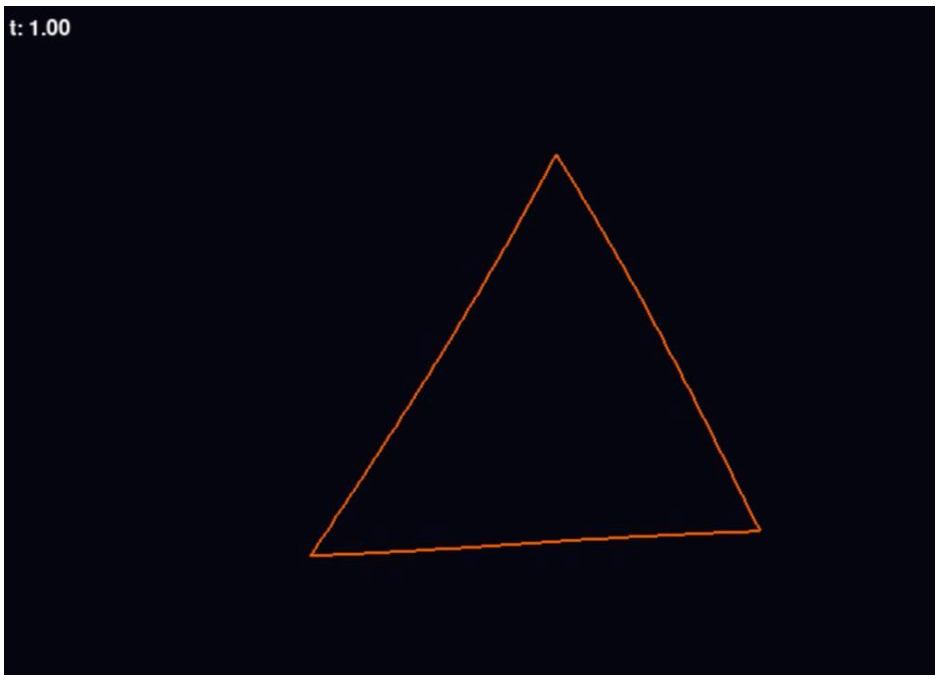
```
    remainder = self.points_count % 3



    for i in range(3):

        x1, y1 = edges[i]

        x2, y2 = edges[(i + 1) % 3]

        n = segment_length + (1 if i < remainder else 0)

        for j in range(n):

            t = j / n

            x = x1 + (x2 - x1) * t

            y = y1 + (y2 - y1) * t

            points.append((x, y))

    return points[:self.points_count]



# Интерполяция цвета

def interpolate_color(self, color1, color2, t):

    return (

        int(color1[0] + (color2[0] - color1[0]) * t),

        int(color1[1] + (color2[1] - color1[1]) * t),

        int(color1[2] + (color2[2] - color1[2]) * t),

    )

# Обновление анимации

def update(self):

    if self.t < 1:

        self.t += self.speed

    for p, target in zip(self.circle_points, self.triangle_points):

        tx, ty = target
```

```
            p.move(tx, ty, self.speed)


    # Рисование
    def draw(self):
        self.screen.fill(self.bg)
        line_color = self.interpolate_color(self.circle_color, self.triangle_color,
self.t)


        for i in range(self.points_count - 1):
            p1 = self.circle_points[i]
            p2 = self.circle_points[i + 1]
            pygame.draw.line(self.screen, line_color, (p1.x, p1.y), (p2.x, p2.y), 2)
        pygame.draw.line(self.screen, line_color,
                         (self.circle_points[-1].x, self.circle_points[-1].y),
                         (self.circle_points[0].x, self.circle_points[0].y), 2)


        # Отображаем текущий параметр t
        t_text = self.font.render(f"t: {self.t:.2f}", True, (255, 255, 255))
        self.screen.blit(t_text, (10, 10))


        pygame.display.flip()


    # Основной цикл
    def run(self):
        while True:
            for event in pygame.event.get():
```

```
                if event.type == pygame.QUIT:
                    pygame.quit()
                    sys.exit()
                elif event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_UP:
                        self.speed = min(self.speed + 0.01, 0.2)
                    elif event.key == pygame.K_DOWN:
                        self.speed = max(self.speed - 0.001, 0.001)
                    elif event.key == pygame.K_r:
                        self.t = 0
                        # возвращаем точки на круг
                        for i, p in enumerate(self.circle_points):
                            p.x, p.y = p.polar_to_cartesian(self.radius)

            self.update()
            self.draw()
            self.clock.tick(self.fps)


# ЗАПУСК
if __name__ == "__main__":
    config = load_config()
    app = CircleToTriangle(config)
    app.run()
```

Визуальное доказательство работоспособности :




t: 1.00