МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Лабораторная работа №7
По дисциплине “**Современные платформы программирования**”

Выполнил:
студент группы ПО-11
Сымоник И.А.
Проверил:
Козик И. Д.

Брест 2025

Цель: освоить возможности языка программирования Python в разработке оконных приложений

Задание 1. Построение графических примитивов и надписей

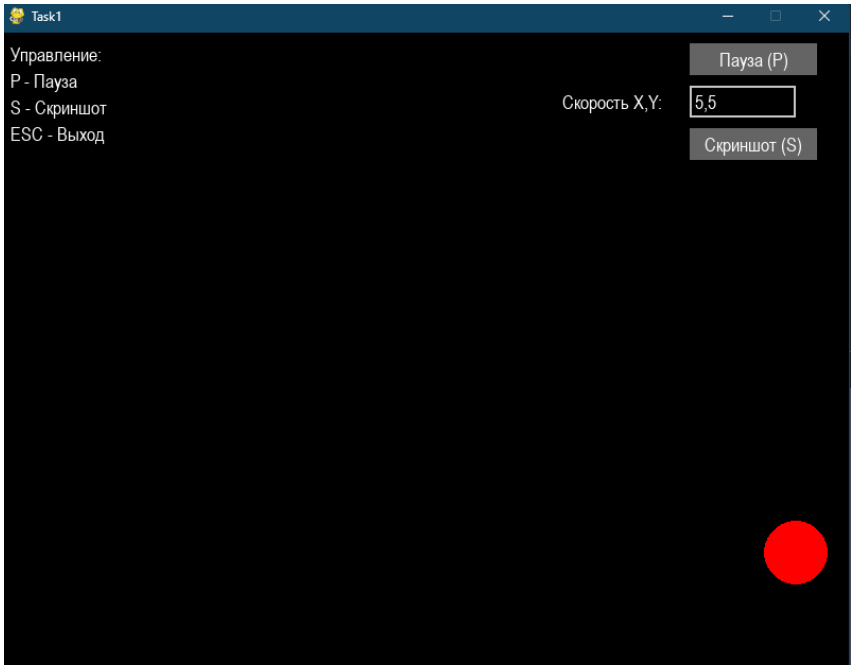
6) Задать движение окружности по форме так, чтобы при касании границы окружность отражалась от нее.

import pygame
import sys
import os
from pygame.locals import *

```
class Circle:
    def __init__(self, x, y, radius, color, speed_x, speed_y):
        self.x = x
        self.y = y
        self.radius = radius
        self.color = color
        self.speed_x = speed_x
        self.speed_y = speed_y
        self.is_paused = False

    def move(self, screen_width, screen_height):
        if not self.is_paused:
            self.x += self.speed_x
            self.y += self.speed_y

            if self.x - self.radius <= 0 or self.x + self.radius >= screen_width:
                self.speed_x = -self.speed_x
                self.x = max(self.radius, min(self.x, screen_width - self.radius))

            if self.y - self.radius <= 0 or self.y + self.radius >= screen_height:
                self.speed_y = -self.speed_y
                self.y = max(self.radius, min(self.y, screen_height - self.radius))

    def draw(self, surface):
        pygame.draw.circle(surface, self.color, (int(self.x), int(self.y)), self.radius)

    def toggle_pause(self):
        self.is_paused = not self.is_paused

    def set_speed(self, speed_x, speed_y):
        if self.speed_x != 0 and speed_x != 0:
            self.speed_x = abs(speed_x) if self.speed_x > 0 else -abs(speed_x)
        else:
            self.speed_x = speed_x

        if self.speed_y != 0 and speed_y != 0:
            self.speed_y = abs(speed_y) if self.speed_y > 0 else -abs(speed_y)
        else:
            self.speed_y = speed_y

class App:
    def __init__(self):
        pygame.init()
        self.width, self.height = 800, 600
        self.screen = pygame.display.set_mode((self.width, self.height))
        pygame.display.set_caption("Task1")

        self.circle = Circle(
            x=self.width//2,
            y=self.height//2,
            radius=30,
            color=(255, 0, 0),
            speed_x=5,
```

```
            speed_y=5
        )

        self.font = pygame.font.SysFont('Arial', 18)

        self.ui_elements = []
        self.create_ui()

        self.speed_input_active = False
        self.speed_input_text = ""
        self.speed_input_rect = pygame.Rect(650, 50, 100, 30)
        self.current_speed_display = "5,5"

        self.screenshot_counter = 1

    def create_ui(self):
        pause_button = {
            'rect': pygame.Rect(650, 10, 120, 30),
            'color': (100, 100, 100),
            'text': 'Пауза (P)',
            'action': self.circle.toggle_pause
        }
        self.ui_elements.append(pause_button)

        screenshot_button = {
            'rect': pygame.Rect(650, 90, 120, 30),
            'color': (100, 100, 100),
            'text': 'Скриншот (S)',
            'action': self.take_screenshot
        }
        self.ui_elements.append(screenshot_button)

    def take_screenshot(self):
        filename = f"screenshot_{self.screenshot_counter}.png"
        pygame.image.save(self.screen, filename)
        print(f"Скриншот сохранен как {filename}")
        self.screenshot_counter += 1

    def format_speed_display(self, speed_x, speed_y):
        def format_speed(s):
            return str(int(s)) if s.is_integer() else str(s)

        return f"{format_speed(abs(speed_x))},{format_speed(abs(speed_y))}"

    def handle_events(self):
        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
                sys.exit()

            elif event.type == KEYDOWN:
                if event.key == K_p:
                    self.circle.toggle_pause()
                elif event.key == K_s:
                    self.take_screenshot()
                elif event.key == K_ESCAPE:
                    pygame.quit()
                    sys.exit()

                if self.speed_input_active:
                    if event.key == K_RETURN:
                        try:
                            speeds = [float(s.strip()) for s in self.speed_input_text.split(',')]
                            if len(speeds) == 2:
                                self.circle.set_speed(speeds[0], speeds[1])
                                self.current_speed_display = self.format_speed_display(speeds[0],
speeds[1])
                        except ValueError:
                            print("Некорректный ввод скорости")
                        self.speed_input_active = False
```

```
                        self.speed_input_text = ""
                    elif event.key == K_BACKSPACE:
                        self.speed_input_text = self.speed_input_text[:-1]
                    elif event.unicode.isdigit() or event.unicode in ',.-':
                        self.speed_input_text += event.unicode

            elif event.type == MOUSEBUTTONDOWN:
                if event.button == 1:
                    for element in self.ui_elements:
                        if element['rect'].collidepoint(event.pos):
                            element['action']()

                    if self.speed_input_rect.collidepoint(event.pos):
                        self.speed_input_active = True
                        self.speed_input_text = ""
                    else:
                        self.speed_input_active = False

    def draw_ui(self):
        for element in self.ui_elements:
            pygame.draw.rect(self.screen, element['color'], element['rect'])
            text_surface = self.font.render(element['text'], True, (255, 255, 255))
            text_rect = text_surface.get_rect(center=element['rect'].center)
            self.screen.blit(text_surface, text_rect)

        color = (255, 255, 255) if self.speed_input_active else (200, 200, 200)
        pygame.draw.rect(self.screen, color, self.speed_input_rect, 2)

        speed_text = self.font.render("Скорость X,Y:", True, (255, 255, 255))
        self.screen.blit(speed_text, (self.speed_input_rect.x - 120, self.speed_input_rect.y +
5))

        display_text = self.speed_input_text if self.speed_input_active else
self.current_speed_display
        text_surface = self.font.render(display_text, True, (255, 255, 255))
        self.screen.blit(text_surface, (self.speed_input_rect.x + 5, self.speed_input_rect.y +
5))

        instructions = [
            "Управление:",
            "P - Пауза",
            "S - Скриншот",
            "ESC - Выход",
        ]

        for i, line in enumerate(instructions):
            text = self.font.render(line, True, (255, 255, 255))
            self.screen.blit(text, (10, 10 + i * 25))

    def run(self):
        clock = pygame.time.Clock()

        while True:
            self.handle_events()

            self.screen.fill((0, 0, 0))

            self.circle.move(self.width, self.height)
            self.circle.draw(self.screen)

            self.draw_ui()

            pygame.display.flip()
            clock.tick(60)

if __name__ == "__main__":
    app = App()
    app.run()
```



Задание 2. Реализовать построение заданного типа фрактала по варианту
6) Склоненное дерево Пифагора (обдуваемое ветром)

```
import pygame
import sys
import math
import os
from pygame.locals import *

class Camera:
    def __init__(self, width, height):
        self.x = width // 2
        self.y = height // 2
        self.zoom = 1.0
        self.drag_start = None

    def world_to_screen(self, point):
        return (
            (point[0] - self.x) * self.zoom + self.x,
            (point[1] - self.y) * self.zoom + self.y
        )

    def screen_to_world(self, point):
        return (
            (point[0] - self.x) / self.zoom + self.x,
            (point[1] - self.y) / self.zoom + self.y
        )

    def begin_drag(self, pos):
        self.drag_start = pos

    def update_drag(self, pos):
        if self.drag_start:
            dx = pos[0] - self.drag_start[0]
            dy = pos[1] - self.drag_start[1]
            self.x -= dx / self.zoom
            self.y -= dy / self.zoom
            self.drag_start = pos

    def end_drag(self):
        self.drag_start = None
```

```
    def zoom_in(self, factor=1.1):
        self.zoom *= factor

    def zoom_out(self, factor=1.1):
        self.zoom /= factor

    def reset(self, width, height):
        self.x = width // 2
        self.y = height // 2
        self.zoom = 1.0

class PythagorasTree:
    def __init__(self):
        pygame.init()
        self.width, self.height = 1200, 800
        self.screen = pygame.display.set_mode((self.width, self.height), pygame.RESIZABLE)
        pygame.display.set_caption("Дерево Пифагора с камерой")

        self.camera = Camera(self.width, self.height)

        self.params = {
            'angle_left': 45,
            'angle_right': 45,
            'length': 150,
            'depth': 10,
            'color_mode': 0,
            'line_width': 1,
            'size_factor': 0.7,
            'wind_effect': 0
        }

        self.base_pos = [self.width // 2, self.height - 50]

        self.font = pygame.font.SysFont('Arial', 16)
        self.ui_elements = []
        self.create_ui()

        self.active_input = None

    def create_ui(self):
        self.ui_elements = []
        groups = [
            {
                'title': "Форма дерева",
                'params': [
                    ('angle_left', "Угол левых ветвей", 0, 90),
                    ('angle_right', "Угол правых ветвей", 0, 90),
                    ('length', "Длина ствола", 50, 300),
                    ('depth', "Глубина ветвления", 1, 15),
                    ('size_factor', "Уменьшение ветвей", 0.5, 0.9),
                    ('wind_effect', "Наклон дерева", -30, 30)
                ],
                'pos': (20, 20)
            },
            {
                'title': "Внешний вид",
                'params': [
                    ('line_width', "Толщина линий", 1, 5),
                    ('color_mode', "Цветовой режим", 0, 1)
                ],
                'pos': (20, 300)
            }
        ]
        for group in groups:
            title = self.font.render(group['title'], True, (255, 255, 255))
            self.ui_elements.append({
                'type': 'group_title',
                'surface': title,
                'rect': pygame.Rect(*group['pos'], 200, 20)
            })
```

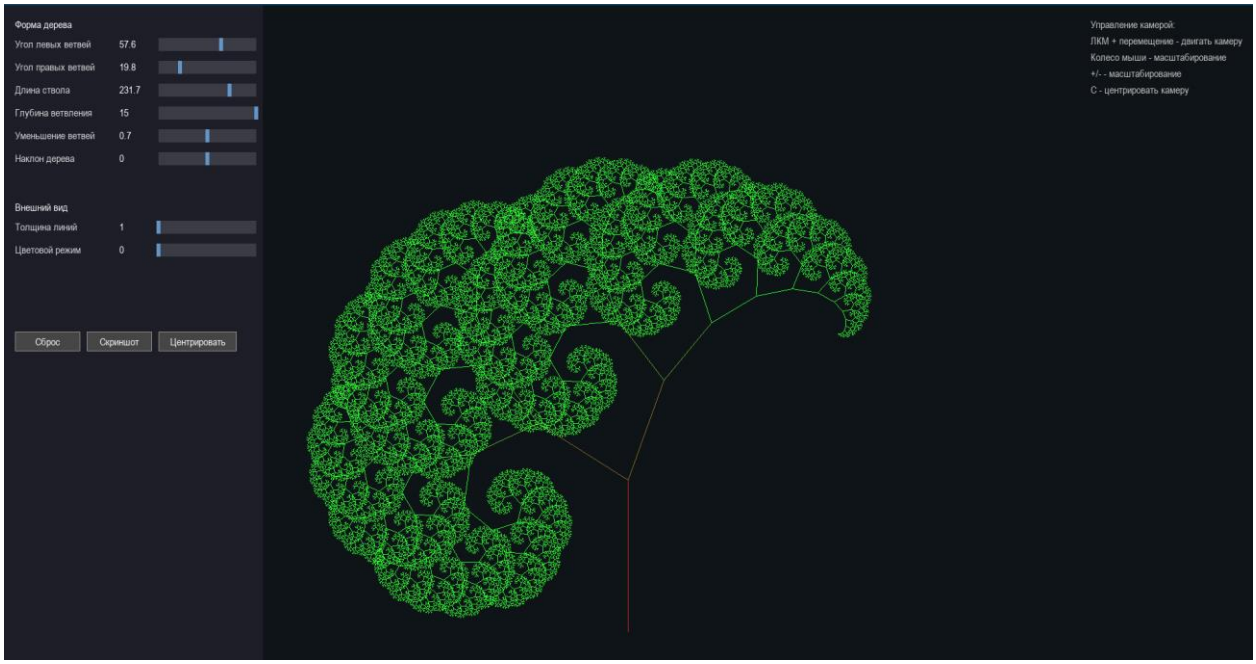
```
            y_offset = group['pos'][1] + 30

            for param in group['params']:
                name, label, min_val, max_val = param
                current_val = self.params[name]

                param_text = self.font.render(label, True, (220, 220, 220))
                self.ui_elements.append({
                    'type': 'param_label',
                    'surface': param_text,
                    'rect': pygame.Rect(group['pos'][0], y_offset, 150, 25)
                })

                value_text = self.font.render(f"{current_val:.1f}", True, (255, 255, 255))
                self.ui_elements.append({
                    'type': 'param_value',
                    'param': name,
                    'surface': value_text,
                    'rect': pygame.Rect(group['pos'][0] + 160, y_offset, 50, 25)
                })

                slider_rect = pygame.Rect(group['pos'][0] + 220, y_offset, 150, 20)
                self.ui_elements.append({
                    'type': 'slider',
                    'param': name,
                    'rect': slider_rect,
                    'min': min_val,
                    'max': max_val,
                    'value': current_val
                })

                y_offset += 35

        buttons = [
            ('reset', "Сброс", (20, 500), (100, 30)),
            ('screenshot', "Скриншот", (130, 500), (100, 30)),
            ('center', "Центрировать", (240, 500), (120, 30))
        ]

        for btn in buttons:
            name, text, pos, size = btn
            rect = pygame.Rect(*pos, *size)
            self.ui_elements.append({
                'type': 'button',
                'name': name,
                'text': text,
                'rect': rect,
                'color': (70, 70, 70),
                'hover_color': (100, 100, 100)
            })

    def draw_tree(self, surface, start_pos, angle, length, depth):
        if depth <= 0 or length < 1:
            return

        screen_start = self.camera.world_to_screen(start_pos)

        angle_rad = math.radians(angle) + math.radians(self.params['wind_effect']) *
(depth/self.params['depth'])
        world_end = (
            start_pos[0] + length * math.cos(angle_rad),
            start_pos[1] - length * math.sin(angle_rad)
        )
        screen_end = self.camera.world_to_screen(world_end)

        if self.params['color_mode'] == 0:
            greenness = min(1, (self.params['depth'] - depth) / 3)
            color = (
                50 + int(150 * (1 - greenness)),
```

```
                50 + int(200 * greenness),
                30 + int(20 * greenness)
            )
        else:
            hue = (depth * 30) % 360
            color = self.hsv_to_rgb(hue, 0.8, 0.7)

        thickness = max(1, int(self.params['line_width'] * (0.5 + depth/self.params['depth']) *
self.camera.zoom))
        pygame.draw.line(surface, color, screen_start, screen_end, thickness)

        if depth > 1:
            new_length = length * self.params['size_factor']

            self.draw_tree(
                surface, world_end,
                angle + self.params['angle_left'],
                new_length, depth - 1)

            self.draw_tree(
                surface, world_end,
                angle - self.params['angle_right'],
                new_length, depth - 1)

    def hsv_to_rgb(self, h, s, v):
        h = h % 360
        c = v * s
        x = c * (1 - abs((h / 60) % 2 - 1))
        m = v - c

        if 0 <= h < 60:
            r, g, b = c, x, 0
        elif 60 <= h < 120:
            r, g, b = x, c, 0
        elif 120 <= h < 180:
            r, g, b = 0, c, x
        elif 180 <= h < 240:
            r, g, b = 0, x, c
        elif 240 <= h < 300:
            r, g, b = x, 0, c
        else:
            r, g, b = c, 0, x

        return (
            int((r + m) * 255),
            int((g + m) * 255),
            int((b + m) * 255)
        )

    def update_ui_values(self):
        for element in self.ui_elements:
            if element['type'] == 'param_value':
                value = self.params[element['param']]
                text = f"{value:.1f}" if isinstance(value, float) else str(value)
                element['surface'] = self.font.render(text, True, (255, 255, 255))

    def handle_events(self):
        mouse_pos = pygame.mouse.get_pos()
        mouse_pressed = pygame.mouse.get_pressed()

        for event in pygame.event.get():
            if event.type == QUIT:
                pygame.quit()
                sys.exit()

            elif event.type == KEYDOWN:
                if event.key == K_s:
                    self.take_screenshot()
                elif event.key == K_r:
                    self.reset_params()
```

```
                elif event.key == K_c:
                    self.center_camera()
                elif event.key == K_ESCAPE:
                    pygame.quit()
                    sys.exit()
                elif event.key == K_PLUS or event.key == K_EQUALS:
                    self.camera.zoom_in()
                elif event.key == K_MINUS:
                    self.camera.zoom_out()

            elif event.type == MOUSEBUTTONDOWN:
                if event.button == 1:
                    ui_clicked = False
                    for element in self.ui_elements:
                        if element['rect'].collidepoint(event.pos):
                            ui_clicked = True
                            if element['type'] == 'button':
                                if element['name'] == 'screenshot':
                                    self.take_screenshot()
                                elif element['name'] == 'reset':
                                    self.reset_params()
                                elif element['name'] == 'center':
                                    self.center_camera()
                            break

                    if not ui_clicked:
                        self.camera.begin_drag(event.pos)

                elif event.button == 4:
                    self.camera.zoom_in()

                elif event.button == 5:
                    self.camera.zoom_out()

            elif event.type == MOUSEBUTTONUP:
                if event.button == 1:
                    self.camera.end_drag()

            elif event.type == MOUSEMOTION:
                if mouse_pressed[0] and self.camera.drag_start:
                    self.camera.update_drag(event.pos)

            elif event.type == VIDEORESIZE:
                self.width, self.height = event.size
                self.screen = pygame.display.set_mode((self.width, self.height),
pygame.RESIZABLE)
                self.base_pos = [self.width // 2, self.height - 50]
                self.camera.reset(self.width, self.height)
                self.create_ui()

        if mouse_pressed[0] and not self.camera.drag_start:
            for element in self.ui_elements:
                if element['type'] == 'slider' and element['rect'].collidepoint(mouse_pos):
                    rel_x = mouse_pos[0] - element['rect'].x
                    percent = min(1, max(0, rel_x / element['rect'].width))
                    new_value = element['min'] + percent * (element['max'] - element['min'])

                    if element['param'] in ['depth', 'color_mode']:
                        new_value = int(round(new_value))

                    self.params[element['param']] = new_value
                    self.update_ui_values()

    def take_screenshot(self):
        tree_surface = pygame.Surface((self.width, self.height), pygame.SRCALPHA)
        tree_surface.fill((0, 0, 0, 0))

        self.draw_tree(
            tree_surface, self.base_pos,
            90, self.params['length'],
```

```
            self.params['depth'])

        counter = 1
        while True:
            filename = f"pythagoras_tree_{counter}.png"
            if not os.path.exists(filename):
                break
            counter += 1

        pygame.image.save(tree_surface, filename)
        print(f"Скриншот сохранен как {filename}")

    def reset_params(self):
        self.params.update({
            'angle_left': 45,
            'angle_right': 45,
            'length': 150,
            'depth': 10,
            'color_mode': 0,
            'line_width': 1,
            'size_factor': 0.7,
            'wind_effect': 0
        })
        self.update_ui_values()

    def center_camera(self):
        self.camera.reset(self.width, self.height)

    def draw_ui(self):
        pygame.draw.rect(self.screen, (30, 30, 40, 220), (0, 0, 400, self.height))

        for element in self.ui_elements:
            if element['type'] in ['group_title', 'param_label', 'param_value']:
                self.screen.blit(element['surface'], element['rect'])
            elif element['type'] == 'slider':
                pygame.draw.rect(self.screen, (60, 60, 70), element['rect'])

                percent = (self.params[element['param']] - element['min']) / (element['max'] -
element['min'])
                slider_pos = element['rect'].x + int(percent * element['rect'].width)
                pygame.draw.rect(
                    self.screen, (100, 150, 200),
                    (slider_pos - 3, element['rect'].y, 6, element['rect'].height))
            elif element['type'] == 'button':
                is_hovered = element['rect'].collidepoint(pygame.mouse.get_pos())

                color = element['hover_color'] if is_hovered else element['color']
                pygame.draw.rect(self.screen, color, element['rect'])
                pygame.draw.rect(self.screen, (150, 150, 150), element['rect'], 1)

                text = self.font.render(element['text'], True, (255, 255, 255))
                text_rect = text.get_rect(center=element['rect'].center)
                self.screen.blit(text, text_rect)

        help_text = [
            "Управление камерой:",
            "ЛКМ + перемещение - двигать камеру",
            "Колесо мыши - масштабирование",
            "+/- - масштабирование",
            "C - центрировать камеру"
        ]

        for i, line in enumerate(help_text):
            text = self.font.render(line, True, (200, 200, 200))
            self.screen.blit(text, (self.width - 250, 20 + i * 25))

    def run(self):
        clock = pygame.time.Clock()

        while True:
```

```
            self.handle_events()

            self.screen.fill((15, 20, 25))

            self.draw_tree(
                self.screen, self.base_pos,
                90, self.params['length'],
                self.params['depth'])

            self.draw_ui()

            pygame.display.flip()
            clock.tick(60)

if __name__ == "__main__":
    app = PythagorasTree()
    app.run()
```



Вывод: освоили возможности языка программирования Python в разработке оконных приложений.