РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ им. П. Лумумба

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Дисциплина: *Компьютерный практикум по моделированию*

Студент: Королёв Иван Андреевич

Группа: НКАбд-04-22

МОСКВА

2024 г.

Цель работы: Научиться работать с классами, объектами, атрибутами, с библиотеками tkinter и random. Написать свою мини-игру

Выполнение работы

Задание 1. Используя библиотеки tkinter и random, необходимо создать окно произвольных размеров и запрограммировать перехват события от мыши.

Листинг программы на языке Python:

In [49]:
```
# Движение мышью

def click(event):
    if event.num == 1:
        print("Левая кнопка мыши")
    elif event.num == 3:
        print("Правая кнопка мыши")

# Окно

root = Tk()

x = int((root.winfo_screenwidth() / 2) - (WIDTH / 2))
y = int((root.winfo_screenheight() / 2) - (HEIGHT / 2))

root.geometry(f"{WIDTH}x{HEIGHT}+{x}+{y}")

root.resizable(False, False)

root.title("Игра с шарами")
root.bind("<Button-1>", click)
root.bind("<Button-3>", click, "+")
root.mainloop()
```

Результаты (численные значения, графики, скриншоты):

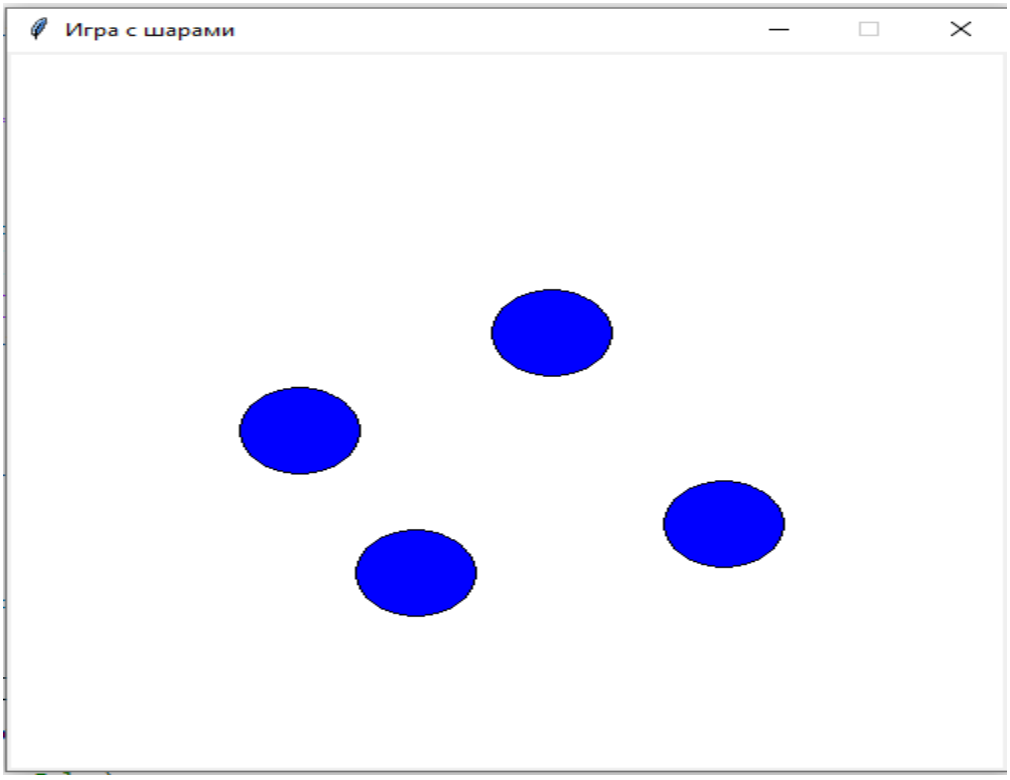
```
Левая кнопка мыши
Левая кнопка мыши
Левая кнопка мыши
Правая кнопка мыши
Правая кнопка мыши
```

Задание 2. Смоделировать основной игровой объект — цветной шарик и заставить его появляться на месте с помощью щелчка мыши.

Листинг программы на языке Python:

In [50]:
```
class Ball:

    def __init__(self, x, y, r, color, speedx=1, speedy=1):
        self.x = x
        self.y = y
        self.r = r
        self.color = color
        self.speedx = speedx
        self.speedy = speedy

    def draw(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=self.color,
        )

# Движение мышью

def click(event):
    main_ball = Ball(event.x, event.y, DELAY, MAIN_BALL_COLOR)
    main_ball.draw()

# Окно

root = Tk()
canvas = Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

x = int((root.winfo_screenwidth() / 2) - (WIDTH / 2))
y = int((root.winfo_screenheight() / 2) - (HEIGHT / 2))

root.geometry(f"{WIDTH}x{HEIGHT}+{x}+{y}")

root.resizable(False, False)

root.title("Игра с шарами")
root.bind("<Button-1>", click)
root.bind("<Button-3>", click, "+")
root.mainloop()
```

Результаты (численные значения, графики, скриншоты):



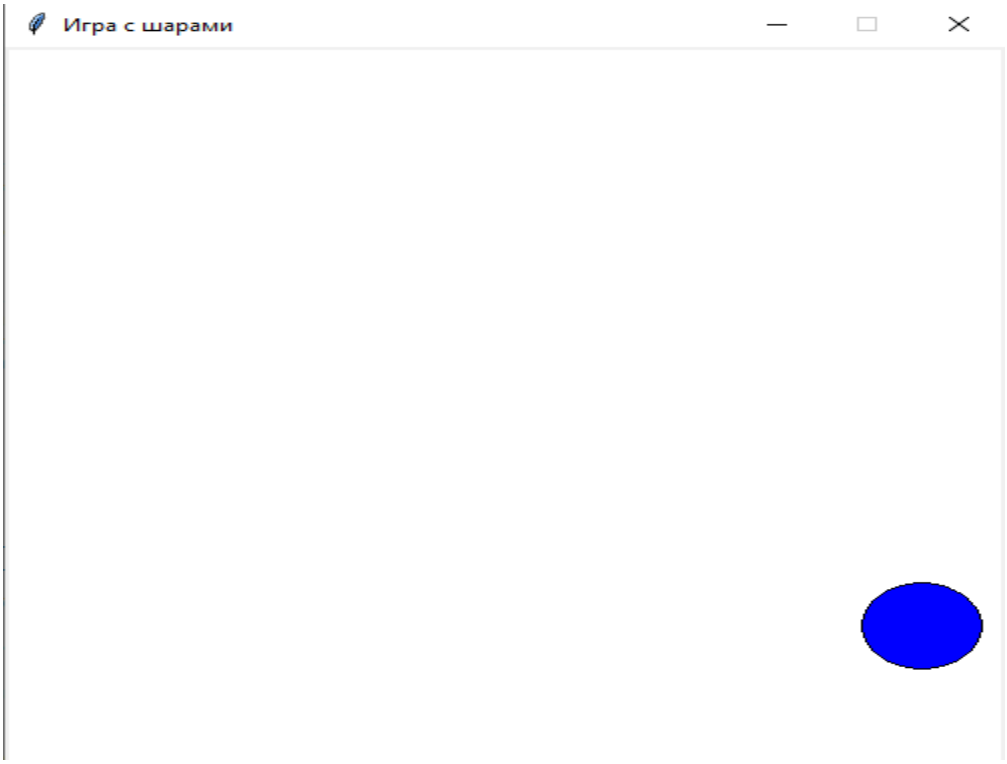
Задание 3. Смоделировать движение шара и отскок от краёв экрана.

Листинг программы на языке Python:

```
class Ball:

    def __init__(self, x, y, r, color, speedx=1, speedy=1):
        self.x = x
        self.y = y
        self.r = r
        self.color = color
        self.speedx = speedx
        self.speedy = speedy

    def draw(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=self.color,
        )
    def hide(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=BG_COLOR,
            outline=BG_COLOR,
        )

    def movement(self):
        if (self.x + self.r + self.speedx >= WIDTH) or (
            self.x - self.r + self.speedx <= 0
        ):
            self.speedx = -self.speedx
        if (self.y + self.r + self.speedy >= HEIGHT) or (
            self.y - self.r + self.speedy <= 0
        ):
            self.speedy = -self.speedy
        self.hide()
        self.x += self.speedx
        self.y += self.speedy
        self.draw()
```

```
# Движение мышью

def click(event):
    global main_ball
    if event.num == 1:
        main_ball = Ball(event.x, event.y, DELAY, MAIN_BALL_COLOR)
        main_ball.draw()
        main_ball.movement()
    else:
        main_ball.hide()

def main():
    global main_ball
    if "main_ball" in globals():
        main_ball.movement()
    root.after(10, main)

# Окно

root = Tk()
canvas = Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

x = int((root.winfo_screenwidth() / 2) - (WIDTH / 2))
y = int((root.winfo_screenheight() / 2) - (HEIGHT / 2))

root.geometry(f"{WIDTH}x{HEIGHT}+{x}+{y}")

root.resizable(False, False)

root.title("Игра с шарами")
root.bind("<Button-1>", click)
root.bind("<Button-3>", click, "+")
if "main_ball" in globals():
    del main_ball
main()
root.mainloop()
```

Результаты (численные значения, графики, скриншоты):



Задание 4. Ввести управление движущимся шариком с помощью мыши

Листинг программы на языке Python:

```
class Ball:

    def __init__(self, x, y, r, color, speedx=1, speedy=1):
        self.x = x
        self.y = y
        self.r = r
        self.color = color
        self.speedx = speedx
        self.speedy = speedy

    def draw(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=self.color,
        )
    def hide(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=BG_COLOR,
            outline=BG_COLOR,
        )

    def movement(self):
        if (self.x + self.r + self.speedx >= WIDTH) or (
            self.x - self.r + self.speedx <= 0
        ):
            self.speedx = -self.speedx
        if (self.y + self.r + self.speedy >= HEIGHT) or (
            self.y - self.r + self.speedy <= 0
        ):
            self.speedy = -self.speedy
        self.hide()
        self.x += self.speedx
        self.y += self.speedy
        self.draw()
```

```
# Движение мышью

def click(event):
    global main_ball
    if event.num == 1:
        if "main_ball" not in globals():  # старт
            main_ball = Ball(
                event.x, event.y, MAIN_BALL_RADIUS, MAIN_BALL_COLOR, INIT_DX, INIT_DY
            )
            if main_ball.x > WIDTH / 2:
                main_ball.speedx = -main_ball.speedx
            if main_ball.y > HEIGHT / 2:
                main_ball.speedy = -main_ball.speedy
            main_ball.draw()
        else:  # turn left
            if main_ball.speedy * main_ball.speedx > 0:
                main_ball.speedy = -main_ball.speedy
            else:
                main_ball.speedx = -main_ball.speedx
    elif event.num == 3:  # right mouse button: turn right
        if main_ball.speedy * main_ball.speedx > 0:
            main_ball.speedx = -main_ball.speedx
        else:
            main_ball.speedy = -main_ball.speedy

def main():
    global main_ball
    if "main_ball" in globals():
        main_ball.movement()
    root.after(10, main)

# Окно

root = Tk()
canvas = Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

x = int((root.winfo_screenwidth() / 2) - (WIDTH / 2))
y = int((root.winfo_screenheight() / 2) - (HEIGHT / 2))

root.geometry(f"{WIDTH}x{HEIGHT}+{x}+{y}")

root.resizable(False, False)

root.title("Игра с шарами")
root.bind("<Button-1>", click)
root.bind("<Button-3>", click, "+")
if "main_ball" in globals():
    del main_ball
main()
root.mainloop()
```

Результаты (численные значения, графики, скриншоты):

Демонстрирую в видео.

Задание 5. Добавить другие неподвижные шары с помощью модуля random, эти шары должен «выбивать» главный шар. Смоделировать события столкновения.

Листинг программы на языке Python:

```
class Ball:

    def __init__(self, x, y, r, color, speedx=1, speedy=1):
        self.x = x
        self.y = y
        self.r = r
        self.color = color
        self.speedx = speedx
        self.speedy = speedy

    def draw(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=self.color,
        )
    def hide(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=BG_COLOR,
            outline=BG_COLOR,
        )

    def is_collision(self, ball):
        a = abs(self.x + self.speedx - ball.x)
        b = abs(self.y + self.speedy - ball.y)
        return (a * a + b * b) ** 0.5 <= self.r + ball.r
```

```
    def movement(self):
        if (self.x + self.r + self.speedx >= WIDTH) or (
            self.x - self.r + self.speedx <= 0
        ):
            self.speedx = -self.speedx
        if (self.y + self.r + self.speedy >= HEIGHT) or (
            self.y - self.r + self.speedy <= 0
        ):
            self.speedy = -self.speedy
        for ball in balls:
            if self.is_collision(ball):
                ball.hide()
                balls.remove(ball)
                self.speedx = -self.speedx
                self.speedy = -self.speedy
        self.hide()
        self.x += self.speedx
        self.y += self.speedy
        self.draw()


# Движение мышью

def click(event):
    global main_ball
    if event.num == 1:
        if "main_ball" not in globals():  # спарт
            main_ball = Ball(
                event.x, event.y, MAIN_BALL_RADIUS, MAIN_BALL_COLOR, INIT_DX, INIT_DY
            )
            if main_ball.x > WIDTH / 2:
                main_ball.speedx = -main_ball.speedx
            if main_ball.y > HEIGHT / 2:
                main_ball.speedy = -main_ball.speedy
            main_ball.draw()
        else:  # turn left
            if main_ball.speedy * main_ball.speedx > 0:
                main_ball.speedy = -main_ball.speedy
            else:
                main_ball.speedx = -main_ball.speedx
    elif event.num == 3:  # right mouse button: turn right
        if main_ball.speedy * main_ball.speedx > 0:
            main_ball.speedx = -main_ball.speedx
        else:
            main_ball.speedy = -main_ball.speedy
```

```
def create_list_of_balls(number):
    lst = []
    while len(lst) < number:
        next_ball = Ball(
            random.choice(range(MAX_RADIUS, WIDTH - MAX_RADIUS)),
            random.choice(range(MAX_RADIUS, HEIGHT - MAX_RADIUS)),
            random.choice(range(MIN_RADIUS, MAX_RADIUS)),
            random.choice(COLORS),
        )
        is_collision = False
        for ball in lst:
            if next_ball.is_collision(ball):
                is_collision = True
                break
        if not is_collision:
            lst.append(next_ball)
            next_ball.draw()
    return lst


def main():
    global main_ball
    if "main_ball" in globals():
        main_ball.movement()
    root.after(10, main)

# Окно

root = Tk()
canvas = Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

x = int((root.winfo_screenwidth() / 2) - (WIDTH / 2))
y = int((root.winfo_screenheight() / 2) - (HEIGHT / 2))

root.geometry(f"{WIDTH}x{HEIGHT}+{x}+{y}")

root.resizable(False, False)

root.title("Игра с шарами")
root.bind("<Button-1>", click)
root.bind("<Button-3>", click, "+")
if "main_ball" in globals():
    del main_ball
balls = create_list_of_balls(NUM_OF_BALLS)
main()
root.mainloop()
```

Результаты (численные значения, графики, скриншоты):

Демонстрирую в видео

Задание 6. Определить «опасные» шары. Если главный шар касается их — наступает проигрыш.

```
class Ball:

    def __init__(self, x, y, r, color, speedx=1, speedy=1):
        self.x = x
        self.y = y
        self.r = r
        self.color = color
        self.speedx = speedx
        self.speedy = speedy

    def draw(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=self.color,
        )
    def hide(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=BG_COLOR,
            outline=BG_COLOR,
        )

    def is_collision(self, ball):
        a = abs(self.x + self.speedx - ball.x)
        b = abs(self.y + self.speedy - ball.y)
        return (a * a + b * b) ** 0.5 <= self.r + ball.r
```

```
    def movement(self):
        if (self.x + self.r + self.speedx >= WIDTH) or (
            self.x - self.r + self.speedx <= 0
        ):
            self.speedx = -self.speedx
        if (self.y + self.r + self.speedy >= HEIGHT) or (
            self.y - self.r + self.speedy <= 0
        ):
            self.speedy = -self.speedy
        for ball in balls:
            if self.is_collision(ball):
                if ball.color != BAD_COLOR:
                    ball.hide()
                    balls.remove(ball)
                    self.speedx = -self.speedx
                    self.speedy = -self.speedy
                else:
                    self.speedx = self.speedy = 0
        self.hide()
        self.x += self.speedx
        self.y += self.speedy
        if self.speedx * self.speedy != 0:
            self.draw()


# Движение мышью

def click(event):
    global main_ball
    if event.num == 1:
        if "main_ball" not in globals():  # спарт
            main_ball = Ball(
                event.x, event.y, MAIN_BALL_RADIUS, MAIN_BALL_COLOR, INIT_DX, INIT_DY
            )
            if main_ball.x > WIDTH / 2:
                main_ball.speedx = -main_ball.speedx
            if main_ball.y > HEIGHT / 2:
                main_ball.speedy = -main_ball.speedy
            main_ball.draw()
        else:  # turn left
            if main_ball.speedy * main_ball.speedx > 0:
                main_ball.speedy = -main_ball.speedy
            else:
                main_ball.speedx = -main_ball.speedx
    elif event.num == 3:  # right mouse button: turn right
        if main_ball.speedy * main_ball.speedx > 0:
            main_ball.speedx = -main_ball.speedx
        else:
            main_ball.speedy = -main_ball.speedy
```

```
def create_list_of_balls(number):
    lst = []
    while len(lst) < number:
        next_ball = Ball(
            random.choice(range(MAX_RADIUS, WIDTH - MAX_RADIUS)),
            random.choice(range(MAX_RADIUS, HEIGHT - MAX_RADIUS)),
            random.choice(range(MIN_RADIUS, MAX_RADIUS)),
            random.choice(COLORS),
        )
        is_collision = False
        for ball in lst:
            if next_ball.is_collision(ball):
                is_collision = True
                break
        if not is_collision:
            lst.append(next_ball)
            next_ball.draw()
    return lst

def count_bad_balls(list_of_balls):
    result = 0
    for ball in list_of_balls:
        if ball.color == BAD_COLOR:
            result += 1
    return result


def main():
    global main_ball
    if "main_ball" in globals():
        main_ball.movement()
        if len(balls) - num_of_bad_balls == 0:
            canvas.create_text(
                WIDTH / 2, HEIGHT / 2, text="YOU WON!", font="Arial 20", fill="lime"
            )
            main_ball.speedx = main_ball.speedy = 0
        elif main_ball.speedx * main_ball.speedy == 0:
            canvas.create_text(
                WIDTH / 2, HEIGHT / 2, text="YOU LOSE!", font="Arial 20", fill="red"
            )
    root.after(DELAY, main)
```

```
# Окно

root = Tk()
canvas = Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()

x = int((root.winfo_screenwidth() / 2) - (WIDTH / 2))
y = int((root.winfo_screenheight() / 2) - (HEIGHT / 2))

root.geometry(f"{WIDTH}x{HEIGHT}+{x}+{y}")

root.resizable(False, False)

root.title("Игра с шарами")
root.bind("<Button-1>", click)
root.bind("<Button-3>", click, "+")
if "main_ball" in globals():
    del main_ball
balls = create_list_of_balls(NUM_OF_BALLS)
num_of_bad_balls = count_bad_balls(balls)
main()
root.mainloop()
```

Результаты (численные значения, графики, скриншоты):

Демонстрирую в видео

Задание 7. Усложняем задачу: смоделировать движение «опасных» шаров.

Итоговый код игры.

```
# Функции и классы для игры

class Ball:

    def __init__(self, x, y, r, color, speedx=1, speedy=1):
        self.x = x
        self.y = y
        self.r = r
        self.color = color
        self.speedx = speedx
        self.speedy = speedy

    def draw(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=self.color,
            outline=self.color if self.color != BAD_COLOR else "black",
        )

    def hide(self):
        canvas.create_oval(
            self.x - self.r,
            self.y - self.r,
            self.x + self.r,
            self.y + self.r,
            fill=BG_COLOR,
            outline=BG_COLOR,
        )

    def is_collision(self, ball):
        a = abs(self.x + self.speedx - ball.x)
        b = abs(self.y + self.speedy - ball.y)
        return (a * a + b * b) ** 0.5 <= self.r + ball.r
```

```
def movement(self):
    if (self.x + self.r + self.speedx >= WIDTH) or (
        self.x - self.r + self.speedx <= 0
    ):
        self.speedx = -self.speedx
    if (self.y + self.r + self.speedy >= HEIGHT) or (
        self.y - self.r + self.speedy <= 0
    ):
        self.speedy = -self.speedy
    for ball in balls:
        if self.is_collision(ball):
            if ball.color != BAD_COLOR:
                ball.hide()
                balls.remove(ball)
                self.speedx = -self.speedx
                self.speedy = -self.speedy
            else:
                self.speedx = self.speedy = 0
    self.hide()
    self.x += self.speedx
    self.y += self.speedy
    if self.speedx * self.speedy != 0:
        self.draw()


def move_bad(self):
    if (self.x + self.r + self.speedx >= WIDTH) or (
        self.x - self.r + self.speedx <= 0
    ):
        self.speedx = -self.speedx
    if (self.y + self.r + self.speedy >= HEIGHT) or (
        self.y - self.r + self.speedy <= 0
    ):
        self.speedy = -self.speedy
    for (
        ball
    ) in (
        balls
    ):
        if ball != self:
            if self.is_collision(ball):
                self.speedx = -self.speedx
                self.speedy = -self.speedy
    self.hide()
    self.x += self.speedx
    self.y += self.speedy
    self.draw()
```

```
def click(event):
    global main_ball
    if event.num == 1:
        if "main_ball" not in globals():  # cmapm
            main_ball = Ball(
                event.x, event.y, MAIN_BALL_RADIUS, MAIN_BALL_COLOR, INIT_DX, INIT_DY
            )
            if main_ball.x > WIDTH / 2:
                main_ball.speedx = -main_ball.speedx
            if main_ball.y > HEIGHT / 2:
                main_ball.speedy = -main_ball.speedy
            main_ball.draw()
        else:  # turn left
            if main_ball.speedy * main_ball.speedx > 0:
                main_ball.speedy = -main_ball.speedy
            else:
                main_ball.speedx = -main_ball.speedx
    elif event.num == 3:  # right mouse button: turn right
        if main_ball.speedy * main_ball.speedx > 0:
            main_ball.speedx = -main_ball.speedx
        else:
            main_ball.speedy = -main_ball.speedy


# count the number of bad balls
def count_bad_balls(list_of_balls):
    result = 0
    for ball in list_of_balls:
        if ball.color == BAD_COLOR:
            result += 1
    return result


def create_list_of_balls(number):
    lst = []
    while len(lst) < number:
        next_ball = Ball(
            random.choice(range(MAX_RADIUS, WIDTH - MAX_RADIUS)),
            random.choice(range(MAX_RADIUS, HEIGHT - MAX_RADIUS)),
            random.choice(range(MIN_RADIUS, MAX_RADIUS)),
            random.choice(COLORS),
        )
        is_collision = False
        for ball in lst:
            if next_ball.is_collision(ball):
                is_collision = True
                break
        if not is_collision:
            lst.append(next_ball)
            next_ball.draw()
    return lst
```

```
def main():
    global main_ball
    if "main_ball" in globals():
        main_ball.movement()
        for ball in balls:
            if ball.color == BAD_COLOR:  # Двигаем только опасные шары
                ball.move_bad()
        if len(balls) - num_of_bad_balls == 0:
            canvas.create_text(
                WIDTH / 2, HEIGHT / 2, text="YOU WON!", font="Arial 20", fill="lime"
            )
            main_ball.speedx = main_ball.speedy = 0
        elif main_ball.speedx * main_ball.speedy == 0:
            canvas.create_text(
                WIDTH / 2, HEIGHT / 2, text="YOU LOSE!", font="Arial 20", fill="red"
            )
    root.after(DELAY, main)


# Tkinter

root = Tk()

x = int((root.winfo_screenwidth() / 2) - (WIDTH / 2))
y = int((root.winfo_screenheight() / 2) - (HEIGHT / 2))

root.geometry(f"{WIDTH}x{HEIGHT}+{x}+{y}")

# Окно

root.title("Игра с шарами")

root.bind("<Button-1>", click)
root.bind("<Button-3>", click, "+")

# Canvas

canvas = Canvas(root, width=WIDTH, height=HEIGHT, bg="white")
canvas.pack()
balls = create_list_of_balls(NUM_OF_BALLS)
num_of_bad_balls = count_bad_balls(balls)
main()
if "main_ball" in globals():
    del main_ball
root.mainloop()
```

Заключение.

Я научился работать с классами, с его объектами, экземплярами и атрибутами, с библиотеками tkinter и random. Написал мини-игру.