Урок 10. Canvas. Идентификаторы, теги и анимация

Изучив размещение геометрических примитивов на экземпляре Canvas, в этом уроке рассмотрим, как можно обращаться к уже созданным фигурам для изменения их свойств, а также создадим анимацию.

В Tkinter существует два способа "пометить" фигуры, размещенные на холсте, – это идентификаторы и теги. Первые всегда уникальны для каждого объекта. Два объекта не могут иметь одни и тот же идентификатор. Теги не уникальны. Группа объектов на холсте может иметь один и тот же тег. Это дает возможность менять свойства всей группы. Отдельно взятая фигура на Canvas может иметь как идентификатор, так и тег.

Идентификаторы

Методы, создающие фигуры на холсте, возвращают численные идентификаторы этих объектов, которые можно присвоить переменным, через которые позднее обращаться к созданным фигурам.

```
from tkinter import *
root = Tk()
c = Canvas(width=300, height=300, bg='white')
c.focus_set()
c.pack()

ball = c.create_oval(140, 140, 160, 160, fill='green')
c.bind('<Up>', lambda event: c.move(ball, 0, -2))
c.bind('<Down>', lambda event: c.move(ball, 0, 2))
c.bind('<Left>', lambda event: c.move(ball, -2, 0))
c.bind('<Right>', lambda event: c.move(ball, 2, 0))

root.mainloop()
```

В данном примере круг двигается по холсту с помощью стрелок на клавиатуре. Когда создавался круг, его идентификатор был присвоен переменной ball. Метод move() объекта Canvas принимает идентификатор и смещение по осям.

С помощью метода itemconfig() можно изменять другие свойства. Метод coords() устанавливает новые координаты фигуры, если они заданы. Если указывается только идентификатор или тег, то coords() возвращает текущие координаты.

```
from tkinter import *
root = Tk()
c = Canvas(width=200, height=200, bg='white')
c.pack()

rect = c.create_rectangle(80, 80, 120, 120, fill='lightgreen')

def inFocus(event):
    c.itemconfig(rect, fill='green', width=2)
    c.coords(rect, 70, 70, 130, 130)
c.bind('<FocusIn>', inFocus)

root.mainloop()
```

Здесь при получении холстом фокуса (нажать Tab) изменится цвет и размер квадрата.

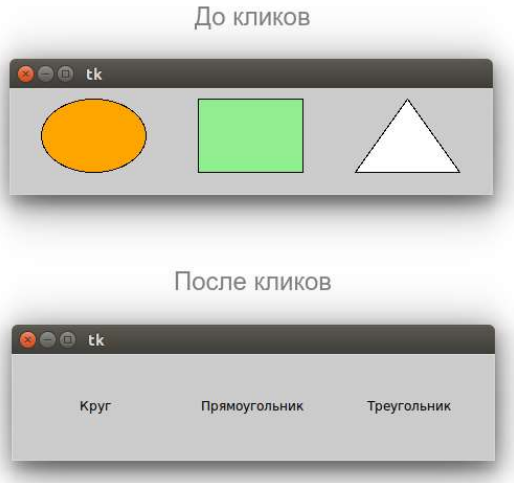
Теги

В отличие от идентификаторов, которые являются уникальными для каждого объекта, один и тот же тег может присваиваться разным объектам. Дальнейшее обращение к такому тегу позволит изменить все объекты, в которых он был указан. В примере ниже эллипс и линия содержат один и тот же тег, а функция color изменяет цвет всех объектов с тегом group1. Обратите внимание, что в отличие от имени идентификатора (переменная), имя тега заключается в кавычки (строковое значение).

```
…
oval = c.create_oval(30,10,130,80,tag="group1")
c.create_line(10,100,450,100,tag="group1")

def color(event):
     c.itemconfig('group1',fill="red",width=3)

c.bind('<Button-3>',color)
…
```

Метод tag_bind() позволяет привязать событие (например, щелчок кнопкой мыши) к определенной фигуре на Canvas. Таким образом, можно реализовать обращение к различным областям холста с помощью одного и того же события. Пример ниже иллюстрирует, как изменения на холсте зависят от того, где произведен клик.

```
from tkinter import *

c = Canvas(width=460, height=100, bg='grey80')
c.pack()

oval = c.create_oval(30, 10, 130, 80, fill="orange")
c.create_rectangle(180, 10, 280, 80,
                   tag="rect", fill="lightgreen")
trian = c.create_polygon(330, 80, 380, 10, 430, 80,
                   fill='white',outline="black")

def oval_func(event):
     c.delete(oval)
     c.create_text(80, 50, text="Круг")
def rect_func(event):
     c.delete("rect")
     c.create_text(230, 50, text="Прямоугольник")
def triangle(event):
     c.delete(trian)
     c.create_text(380, 50, text="Треугольник")

c.tag_bind(oval, '<Button-1>', oval_func)
c.tag_bind("rect", '<Button-1>', rect_func)
c.tag_bind(trian, '<Button-1>', triangle)

mainloop()
```



Метод delete() удаляет объект. Если нужно очистить холст, то вместо идентификаторов или тегов используется константа ALL.

Практическая работа. Анимация в tkinter

В данной программе создается анимация круга, который движется от левой границы холста до правой:

```
from tkinter import *

root = Tk()
c = Canvas(root, width=300, height=200, bg="white")
c.pack()

ball = c.create_oval(0, 100, 40, 140, fill='green')

def motion():
    c.move(ball, 1, 0)
    if c.coords(ball)[2] < 300:
        root.after(10, motion)

motion()

root.mainloop()
```

Выражение `c.coords(ball)` возвращает список текущих координат объекта (в данном случае это `ball`). Третий элемент списка соответствует его второй координате *x*.

Метод after() вызывает функцию, переданную вторым аргументом, через количество миллисекунд, указанных первым аргументом.

Изучите приведенную программу и самостоятельно запрограммируйте постепенное движение фигуры в ту точку холста, где пользователь кликает левой кнопкой мыши. Координаты события хранятся в его атрибутах *x* и *y* (`event.x`, `event.y`).