

Tkinter Canvas

The **Canvas** widget in the **Tkinter** module is one of the most powerful widgets used for **drawing graphics and creating custom graphical user interfaces (GUIs)**.

It provides a blank rectangular area where we can draw different types of objects such as **lines, rectangles, circles, ovals, polygons, arcs, text, images**, and even place other Tkinter widgets like **buttons, labels, and entry boxes**.

Unlike widgets such as `Label` or `Button`, which display predefined content, the **Canvas** allows programmers to create and manipulate graphical objects dynamically.

The **Canvas** widget works on a **coordinate system**, where the **top-left corner** of the canvas is the origin $(0, 0)$. The **x-coordinate** increases from **left to right**, while the **y-coordinate** increases from **top to bottom**.



The most commonly used `Canvas` methods are:

1. `create_line()`

The `create_line()` method is used to draw one or more straight lines on the canvas. You can specify the starting and ending coordinates, line color, width, and style.

Syntax

```
canvas.create_line(x1, y1, x2, y2, options)
```

Parameters

Parameter	Description
<code>x1, y1</code>	Starting point of the line
<code>x2, y2</code>	Ending point of the line
<code>fill</code>	Color of the line
<code>width</code>	Thickness of the line
<code>dash</code>	Creates a dashed line

Example

```
import tkinter as tk

root = tk.Tk()

canvas = tk.Canvas(root, width=300, height=200)
canvas.pack()

canvas.create_line(20, 30, 250, 30, fill="blue", width=3)

root.mainloop()
```

2. create_rectangle()

The `create_rectangle()` method draws a rectangle or square using two opposite corner coordinates.

Syntax

```
canvas.create_rectangle(x1, y1, x2, y2, options)
```

Parameters

Parameter	Description
x1, y1	Top-left corner
x2, y2	Bottom-right corner
fill	Interior color
outline	Border color
width	Border thickness

Example

```
canvas.create_rectangle(
    50, 50,
    180, 120,
    fill="yellow",
    outline="black",
    width=2
)
```

3. create_oval()

The `create_oval()` method draws an oval inside an imaginary rectangle. If the width and height are equal, it creates a circle.

Syntax

```
canvas.create_oval(x1, y1, x2, y2, options)
```

Parameters

Parameter	Description
x1, y1	Top-left of bounding rectangle
x2, y2	Bottom-right of bounding rectangle
fill	Fill color
outline	Border color

Example

```
canvas.create_oval(  
    60, 60,  
    180, 140,  
    fill="green",  
    outline="black"  
)
```

4. create_arc()

The `create_arc()` method draws part of an oval. It can display only the arc, a chord, or a pie slice.

Syntax

```
canvas.create_arc(x1, y1, x2, y2,  
                 start=angle,  
                 extent=angle,  
                 style=tk.ARC)
```

Styles

Style	Description
tk.ARC	Draws only the curved arc
tk.CHORD	Arc with a straight line joining the ends
tk.PIESLICE	Pie slice with center connected

Example

```
canvas.create_arc(  
    50, 50,  
    180, 180,  
    start=0,  
    extent=90,  
    style=tk.PIESLICE,  
    fill="orange"  
)
```

5. create_polygon()

The `create_polygon()` method draws a closed shape using multiple coordinate points.

Syntax

```
canvas.create_polygon(  
    x1, y1,  
    x2, y2,  
    x3, y3,  
    ...,  
    options  
)
```

Parameters

Parameter	Description
Coordinates	Vertices of the polygon
fill	Fill color
outline	Border color

Example

```
canvas.create_polygon(  
    150, 30,  
    220, 120,  
    80, 120,  
    fill="pink",  
    outline="black"  
)
```

6. create_text()

The `create_text()` method displays text at a specified position on the canvas.

Syntax

```
canvas.create_text(  
    x,  
    y,  
    text="message",  
    font=("Arial", 14),  
    fill="blue"  
)
```

Parameters

Parameter	Description
x, y	Position of text
text	Text to display
font	Font family, size, and style

Parameter	Description
fill	Text color

Example

```
canvas.create_text(
    150, 100,
    text="Welcome",
    font=("Arial", 18, "bold"),
    fill="red"
)
```

7. create_image()

The `create_image()` method displays an image on the canvas. The image should be loaded first using `PhotoImage` or another compatible image class.

Syntax

```
photo = tk.PhotoImage(file="image.png")

canvas.create_image(
    x,
    y,
    image=photo
)
```

Example

```
photo = tk.PhotoImage(file="flower.png")

canvas.create_image(
    150,
    120,
    image=photo
)
```

8. create_window()

The `create_window()` method places another Tkinter widget (such as a `Button`, `Entry`, or `Label`) inside the canvas.

Syntax

```
canvas.create_window(
    x,
    y,
    window=widget
)
```

Example

```
btn = tk.Button(root, text="Click")

canvas.create_window(
    150,
    100,
    window=btn
)
```

9. delete()

The `delete()` method removes one or more objects from the canvas. You can delete a specific item using its ID or remove all items using "all".

Syntax

```
canvas.delete(item)

# or

canvas.delete("all")
```

Example

```
line = canvas.create_line(10, 20, 150, 20)

canvas.delete(line)

# Delete everything
canvas.delete("all")
```

10. move()

The `move()` method moves a canvas object by a specified horizontal (`dx`) and vertical (`dy`) distance.

Syntax

```
canvas.move(item, dx, dy)
```

Parameters

Parameter	Description
item	Object ID
dx	Horizontal movement
dy	Vertical movement

Example

```
rect = canvas.create_rectangle(50, 50, 150, 120)
```

```
canvas.move(rect, 80, 40)
```

11. coords()

The `coords()` method returns or changes the coordinates of a canvas object.

Syntax

```
canvas.coords(item)

# Change coordinates

canvas.coords(item, x1, y1, x2, y2)
```

Example

```
rect = canvas.create_rectangle(20, 20, 80, 80)

canvas.coords(rect, 100, 100, 200, 180)
```

12. itemconfig()

The `itemconfig()` (or `itemconfigure()`) method changes the properties of an existing canvas item, such as its color, text, font, width, or outline, without recreating it.

Syntax

```
canvas.itemconfig(item, option=value)
```

Common Options

Option	Description
fill	Changes the fill or text color
outline	Changes the border color
text	Changes displayed text
font	Changes font
width	Changes line or border thickness

Example

```
text = canvas.create_text(
    100, 50,
    text="Hello",
    fill="blue"
)

# Change the text and color
canvas.itemconfig(
    text,
```

```

    text="Welcome",
    fill="red",
    font=("Arial", 18, "bold")
)

```

Summary Table

Method	Purpose
<code>create_line()</code>	Draws straight lines
<code>create_rectangle()</code>	Draws rectangles and squares
<code>create_oval()</code>	Draws ovals and circles
<code>create_arc()</code>	Draws arcs and pie slices
<code>create_polygon()</code>	Draws polygons such as triangles and pentagons
<code>create_text()</code>	Displays text on the canvas
<code>create_image()</code>	Displays an image
<code>create_window()</code>	Places Tkinter widgets inside the canvas
<code>delete()</code>	Deletes one or more canvas items
<code>move()</code>	Moves canvas objects
<code>coords()</code>	Gets or changes object coordinates
<code>itemconfig()</code>	Modifies the properties of an existing canvas item

These methods form the core of the Tkinter `Canvas` widget and are widely used to create graphical user interfaces, simple drawings, diagrams, animations, and interactive applications.

Full Example:

```

# Import the tkinter module
import tkinter as tk

# Create the main application window
root = tk.Tk()

# Set the title of the window
root.title("Tkinter Canvas Example")

# Set the window size
root.geometry("700x500")

# -----
# Create a Canvas widget
# Parameters:
# root      -> Parent window
# width     -> Width of canvas
# height    -> Height of canvas
# bg        -> Background color
# -----
canvas = tk.Canvas(root, width=650, height=400, bg="white")

# Display the canvas in the window
canvas.pack(pady=20)

# -----

```

```

# Draw a Line
# create_line(x1, y1, x2, y2, options)
# -----
canvas.create_line(
    20, 20,          # Starting point (x1, y1)
    250, 20,         # Ending point (x2, y2)
    fill="blue",     # Line color
    width=3          # Line thickness
)

# -----
# Draw a Rectangle
# create_rectangle(x1, y1, x2, y2)
# -----
canvas.create_rectangle(
    50, 50,
    200, 150,
    fill="lightblue",
    outline="black",
    width=2
)

# -----
# Draw an Oval
# create_oval(x1, y1, x2, y2)
# -----
canvas.create_oval(
    250, 50,
    400, 150,
    fill="yellow",
    outline="red",
    width=2
)

# -----
# Draw a Circle
# A circle is simply an oval with equal width and height.
# -----
canvas.create_oval(
    450, 50,
    550, 150,
    fill="green",
    outline="darkgreen",
    width=3
)

# -----
# Draw an Arc
# style=tk.ARC      -> Only arc
# style=tk.CHORD    -> Arc with straight line
# style=tk.PIESLICE -> Pie slice
# -----
canvas.create_arc(
    50, 180,
    170, 300,
    start=0,
    extent=120,
    style=tk.CHORD,

```

```

        fill="orange"
    )

# -----
# Draw a Polygon
# Used to create triangles and other shapes
# -----
canvas.create_polygon(
    250, 250,
    320, 180,
    390, 250,
    350, 320,
    290, 320,
    fill="pink",
    outline="black"
)

# -----
# Display Text
# create_text(x, y, options)
# -----
canvas.create_text(
    500, 250,
    text="Tkinter Canvas",
    font=("Arial", 18, "bold"),
    fill="purple"
)

# -----
# Draw a Dashed Line
# -----
canvas.create_line(
    450, 300,
    620, 350,
    dash=(5, 3),
    fill="red",
    width=2
)

# -----
# Draw a Point (Small Filled Circle)
# -----
canvas.create_oval(
    600, 100,
    605, 105,
    fill="black"
)

# Start the Tkinter event loop
root.mainloop()

```