tkinter.Canvas リファレンス

概要

Canvasウィジェットは、グラフィックス描画のためのウィジェットです。線、矩形、楕円、テキスト、画像などを描画でき、イベント処理やアニメーションも可能です。ゲーム開発、データ可視化、図形エディタなどの用途に適しています。

基本的な使用方法

```
import tkinter as tk

root = tk.Tk()
canvas = tk.Canvas(root, width=400, height=300, bg='white')
canvas.pack()

# 図形を描画
canvas.create_line(0, 0, 400, 300, fill='red', width=3)
canvas.create_rectangle(50, 50, 150, 100, fill='blue', outline='black')
canvas.create_oval(200, 100, 300, 200, fill='yellow')

root.mainloop()
```

主要なオプション

オプション	説明	デフォルト値	例
`width`	キャンバスの幅	400	`width=500`
`height`	キャンバスの高さ	300	`height=400`
`bg`	背景色	SystemButtonFace	`bg='white'`
`scrollregion`	スクロール可能な領域	None	`scrollregion=(0,0,800,600)`
`highlightthickness`	フォーカス時の枠線の太さ	2	`highlightthickness=0`
`relief`	境界線のスタイル	'flat'	`relief='sunken'`
`borderwidth`	境界線の幅	2	`borderwidth=1`
`cursor`	マウスカーソルの形状	''	`cursor='crosshair'`

主要なメソッド

メソッド	説明	例
`create_line(x1,y1,x2,y2,**options)`	線を描画	`create_line(0,0,100,100,fill='red')`
`create_rectangle(x1,y1,x2,y2,**options)`	矩形を描画	`create_rectangle(10,10,90,60,fill='blue')`
`create_oval(x1,y1,x2,y2,**options)`	楕円を描画	`create_oval(20,20,80,60,fill='green')`
`create_polygon(x1,y1,x2,y2,...,**options)`	多角形を描画	`create_polygon(10,10,50,5,90,20,fill='red')`
`create_text(x,y,**options)`	テキストを描画	`create_text(50,50,text='Hello',font=('Arial',12))`
`create_image(x,y,**options)`	画像を描画	`create_image(50,50,image=photo)`
`create_arc(x1,y1,x2,y2,**options)`	弧を描画	`create_arc(10,10,90,90,start=0,extent=90)`
`delete(item_id)`	アイテムを削除	`delete(item_id)`
`coords(item_id, x1,y1,x2,y2,...)`	アイテムの座標を変更	`coords(rect_id, 20,20,100,80)`
`itemconfig(item_id, **options)`	アイテムの設定を変更	`itemconfig(rect_id, fill='red')`
`move(item_id, dx, dy)`	アイテムを移動	`move(rect_id, 10, 5)`
`find_overlapping(x1,y1,x2,y2)`	指定領域と重複するアイテムを検索	`find_overlapping(0,0,50,50)`
`find_closest(x, y)`	指定座標に最も近いアイテムを検索	`find_closest(event.x, event.y)`
`bbox(item_id)`	アイテムの境界ボックスを取得	`bbox(rect_id)`
`tag_bind(tag, event, callback)`	タグにイベントをバインド	`tag_bind('clickable', '<Button-1>', on_click)`

実用的な例

1. 基本的な図形描画

```
import tkinter as tk

class DrawingApp:
    def __init__(self, root):
        self.root = root
        self.root.title('図形描画アプリ')

        # キャンバス作成
        self.canvas = tk.Canvas(root, width=500, height=400, bg='white')
        self.canvas.pack(padx=10, pady=10)

        # 各種図形を描画
        self.draw_shapes()

    def draw_shapes(self):
        # 線
        self.canvas.create_line(50, 50, 200, 100, fill='red', width=3)

        # 矩形
        self.canvas.create_rectangle(50, 120, 150, 200,
                                     fill='lightblue', outline='blue', width=2)

        # 楕円
        self.canvas.create_oval(200, 120, 300, 200,
                                fill='lightgreen', outline='green', width=2)

        # 多角形（三角形）
        self.canvas.create_polygon(350, 200, 400, 120, 450, 200,
                                   fill='lightyellow', outline='orange', width=2)

        # テキスト
        self.canvas.create_text(250, 50, text='図形描画の例',
                                font=('Arial', 16, 'bold'), fill='darkblue')

root = tk.Tk()
app = DrawingApp(root)
root.mainloop()
```

2. インタラクティブな描画アプリ

```
import tkinter as tk

class InteractiveCanvas:
    def __init__(self, root):
        self.root = root
        self.root.title('インタラクティブキャンバス')

        # コントロールフレーム
        control_frame = tk.Frame(root)
        control_frame.pack(pady=5)

        # 描画モード選択
        self.mode = tk.StringVar(value='rectangle')
        tk.Label(control_frame, text='描画モード:').pack(side=tk.LEFT)
        modes = [('矩形', 'rectangle'), ('楕円', 'oval'), ('線', 'line')]
        for text, mode in modes:
            tk.Radiobutton(control_frame, text=text, variable=self.mode,
                           value=mode).pack(side=tk.LEFT)

        # 色選択
        self.color = tk.StringVar(value='blue')
        tk.Label(control_frame, text='色:').pack(side=tk.LEFT, padx=(20,0))
        colors = [('青', 'blue'), ('赤', 'red'), ('緑', 'green')]
        for text, color in colors:
            tk.Radiobutton(control_frame, text=text, variable=self.color,
                           value=color).pack(side=tk.LEFT)

        # クリアボタン
        tk.Button(control_frame, text='クリア',
                  command=self.clear_canvas).pack(side=tk.LEFT, padx=(20,0))

        # キャンバス
        self.canvas = tk.Canvas(root, width=600, height=400, bg='white',
                                relief=tk.SUNKEN, borderwidth=2)
        self.canvas.pack(padx=10, pady=10)

        # イベントバインド
        self.canvas.bind('<Button-1>', self.start_draw)
        self.canvas.bind('<B1-Motion>', self.draw_motion)
        self.canvas.bind('<ButtonRelease-1>', self.end_draw)

        self.start_x = self.start_y = 0
        self.current_item = None

    def start_draw(self, event):
        self.start_x, self.start_y = event.x, event.y
        mode = self.mode.get()
        color = self.color.get()
```

```
        if mode == 'rectangle':
            self.current_item = self.canvas.create_rectangle(
                self.start_x, self.start_y, self.start_x, self.start_y,
                outline=color, width=2)
        elif mode == 'oval':
            self.current_item = self.canvas.create_oval(
                self.start_x, self.start_y, self.start_x, self.start_y,
                outline=color, width=2)
        elif mode == 'line':
            self.current_item = self.canvas.create_line(
                self.start_x, self.start_y, self.start_x, self.start_y,
                fill=color, width=2)

    def draw_motion(self, event):
        if self.current_item:
            self.canvas.coords(self.current_item,
                               self.start_x, self.start_y, event.x, event.y)

    def end_draw(self, event):
        self.current_item = None

    def clear_canvas(self):
        self.canvas.delete('all')

root = tk.Tk()
app = InteractiveCanvas(root)
root.mainloop()
```

3. アニメーション例

```
 import tkinter as tk
import math

class AnimationDemo:
    def __init__(self, root):
        self.root = root
        self.root.title('アニメーションデモ')

        # キャンバス
        self.canvas = tk.Canvas(root, width=500, height=300, bg='black')
        self.canvas.pack(padx=10, pady=10)

        # アニメーション用のオブジェクト
        self.ball = self.canvas.create_oval(10, 10, 30, 30, fill='red')
        self.x = 250
        self.y = 150
        self.angle = 0

        # コントロール
        control_frame = tk.Frame(root)
        control_frame.pack()

        self.is_running = False
        self.start_button = tk.Button(control_frame, text='開始',
                                      command=self.toggle_animation)
        self.start_button.pack(side=tk.LEFT, padx=5)

        tk.Button(control_frame, text='リセット',
                  command=self.reset_animation).pack(side=tk.LEFT, padx=5)

        # アニメーション開始
        self.animate()

    def animate(self):
        if self.is_running:
            # 円運動
            radius = 100
            self.x = 250 + radius * math.cos(self.angle)
            self.y = 150 + radius * math.sin(self.angle)
            self.angle += 0.1

            # ボールの位置を更新
            self.canvas.coords(self.ball, self.x-10, self.y-10,
                               self.x+10, self.y+10)

        # 次のフレームをスケジュール
        self.root.after(50, self.animate)

    def toggle_animation(self):
        self.is_running = not self.is_running
        self.start_button.config(text='停止' if self.is_running else '開始')

    def reset_animation(self):
        self.is_running = False
        self.angle = 0
        self.x = 250
        self.y = 150
        self.canvas.coords(self.ball, self.x-10, self.y-10, self.x+10, self.y+10)
        self.start_button.config(text='開始')

root = tk.Tk()
app = AnimationDemo(root)
root.mainloop()
```

4. 画像表示とスクロール

```
 import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk

class ImageViewer:
    def __init__(self, root):
        self.root = root
        self.root.title('画像ビューア')

        # メニュー
        menubar = tk.Menu(root)
        root.config(menu=menubar)

        file_menu = tk.Menu(menubar, tearoff=0)
        menubar.add_cascade(label='ファイル', menu=file_menu)
        file_menu.add_command(label='画像を開く', command=self.open_image)

        # スクロール付きキャンバス
        canvas_frame = tk.Frame(root)
        canvas_frame.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

        self.canvas = tk.Canvas(canvas_frame, bg='gray90')

        # スクロールバー
        v_scrollbar = tk.Scrollbar(canvas_frame, orient=tk.VERTICAL,
                                   command=self.canvas.yview)
        h_scrollbar = tk.Scrollbar(canvas_frame, orient=tk.HORIZONTAL,
                                   command=self.canvas.xview)

        self.canvas.configure(yscrollcommand=v_scrollbar.set,
                              xscrollcommand=h_scrollbar.set)

        # レイアウト
        self.canvas.grid(row=0, column=0, sticky='nsew')
        v_scrollbar.grid(row=0, column=1, sticky='ns')
        h_scrollbar.grid(row=1, column=0, sticky='ew')

        canvas_frame.grid_rowconfigure(0, weight=1)
        canvas_frame.grid_columnconfigure(0, weight=1)

        self.image_id = None

    def open_image(self):
        file_path = filedialog.askopenfilename(
            filetypes=[('画像ファイル', '*.png *.jpg *.jpeg *.gif *.bmp')])

        if file_path:
            try:
                # 画像を読み込み
                image = Image.open(file_path)
                self.photo = ImageTk.PhotoImage(image)

                # 既存の画像を削除
                if self.image_id:
                    self.canvas.delete(self.image_id)

                # 新しい画像を表示
                self.image_id = self.canvas.create_image(0, 0, anchor=tk.NW,
                                                         image=self.photo)

                # スクロール領域を設定
                self.canvas.configure(scrollregion=self.canvas.bbox('all'))

            except Exception as e:
                tk.messagebox.showerror('エラー', f'画像を開けませんでした: {e}')

root = tk.Tk()
app = ImageViewer(root)
root.mainloop()
```

ベストプラクティス

項目	説明	例
座標系の理解	左上が(0,0)、右下に向かって増加	`create_line(0, 0, 100, 100)`
アイテムIDの管理	描画したアイテムのIDを保存して後で操作	`self.rect_id = canvas.create_rectangle(...)`
タグの活用	複数のアイテムをグループ化	`create_rectangle(..., tags='group1')`
イベント処理	マウスやキーボードイベントを適切に処理	`canvas.bind('<Button-1>', self.on_click)`
パフォーマンス	大量のアイテムを扱う場合は定期的にdeleteで削除	`canvas.delete('temporary_items')`
座標変換	ウィンドウサイズに応じた座標変換を実装	スケーリング関数を作成
アニメーション	`after()` メソッドを使用してスムーズなアニメーション	`root.after(50, self.animate)`
メモリ管理	画像リファレンスを保持してガベージコレクションを防ぐ	`self.photo = ImageTk.PhotoImage(...)`

描画オプション

共通オプション

オプション	説明	例
`fill`	塗りつぶし色	`fill='red'`
`outline`	輪郭線の色	`outline='black'`
`width`	線の太さ	`width=3`
`tags`	タグ（グループ化用）	`tags='group1'`
`state`	状態（normal/disabled/hidden）	`state='disabled'`

テキスト専用オプション

オプション	説明	例
`text`	表示テキスト	`text='Hello World'`
`font`	フォント	`font=('Arial', 12, 'bold')`
`anchor`	アンカー位置	`anchor='nw'`
`justify`	行揃え	`justify='center'`

参考リンク

- Python tkinter Canvas公式ドキュメント
- tkinter Canvas チュートリアル
- Canvas座標系について