Computer Animation Lab: 03

Contents

Animating Image in Tkinter

In this task, we will animate a car moving along x-axis. We will use the UP button to increase the speed of the animation, and the DOWN button to decrease the speed of the animation.

Car.py

```
from tkinter import *

class MainGUI:
    def __init__(self):
        # Create the main window
        window = Tk()

        # Set dimensions for the canvas
        self.w = 600  # Width
        self.h = 250  # Height

        # Create a canvas for drawing
        self.canvas = Canvas(window, width=self.w, height=self.h, bg='white')

        # Bind keyboard events to increase and decrease speed
        self.canvas.bind("<Up>", self.incSpeed)
        self.canvas.bind("<Down>", self.decSpeed)

        # Pack the canvas into the window
        self.canvas.pack()

        # Load and scale the car image
        self.car = PhotoImage(file="car1.gif").zoom(2)

        # Initial x position of the car
        x = 0

        # Create the car image on the canvas
        self.canvas.create_image(x, self.h/2, image=self.car, tags="car")

        # Set focus to the canvas to capture key events
        self.canvas.focus_set()

        # Set initial speed and sleep time
        self.dx = 10  # Speed of the car
        self.sleep = 100  # Delay in milliseconds
```

```
        # Start the animation loop
        while True:
            # Move the car
            self.canvas.move("car", self.dx, 0)
            self.canvas.after(self.sleep)  # Wait before the next frame
            self.canvas.update()  # Update the canvas

            # Update x position or reset if it goes out of bounds
            if x < self.w + 65:
                x += self.dx
            else:
                x = 0  # Reset position
                self.canvas.delete("car")  # Remove old car image
                self.canvas.create_image(x, 125, image=self.car, tags="car")  # Create new car image

        # Start the Tkinter main loop
        window.mainloop()

    def incSpeed(self, event):
        # Increase the speed of the car, capped at 40
        if self.dx < 40:
            self.dx += 5

    def decSpeed(self, event):
        # Decrease the speed of the car, with a minimum speed of 5
        if self.dx > 5:
            self.dx -= 5


# Create an instance of the MainGUI class to run the application
MainGUI()
```

Step-by-step code

- Import tkinter library and create a new class with $\_\_init\_\_$ method.

```
from tkinter import *

class MainGUI:
    def __init__(self):
```

- Inside $\_\_init\_\_$, initialize the main tkinter window.
- Create a canvas object with width = 600 and height = 250 and white background.

```
window = Tk()
self.w = 600
self.h = 250
self.canvas = Canvas(window, width=self.w, height=self.h,
bg="white")
```

- Use the $bind$ function in the Canvas class to allow the application listen for events of the UP and DOWN buttons.
    - The UP button increases the speed, bind the function $incSpeed$.
    - The DOWN button decreases the speed, bind the function $decSpeed$.
- Place the canvas object to the main window using $pack()$ function.

```
self.canvas.bind("<Up>", self.incSpeed)
self.canvas.bind("<Down>", self.decSpeed)
self.canvas.pack()
```

- Create a photo image object that holds the image of the car to display on the canvas.
    - Zoom in the image if required.

```
self.car = PhotoImage(file="car1.gif")
self.car = self.car.zoom(2)
```

- From the canvas object, create an image object with x = 0 and y = middle of the window.
- Define the distance to move the image.

```
x = 0
self.canvas.create_image(x, self.h/2, image=self.car, tags="car")
self.dx = 10
```

- Define the infinite loop to run the application thread.
- Inside the loop, use $move$ function to move the car a distance dx in x-axis direction.
- Make the thread sleeps for 100 ms.
- Update the canvas.

```
while True:
    self.canvas.move("car", self.dx, 0)
    self.canvas.after(100)
    self.canvas.update()
```

- Check for the car boundaries inside the loop:
    - If the car's position is less than the width of canvas, continue increasing x-position.
    - If the car's position is greater than the width of the canvas, set x to 0 and redraw the car.

```
if x < self.w+ 65:
    x += self.dx
else:
    x = 0
    self.canvas.delete("car")
    self.canvas.create_image(x, self.h/2, image=self.car,
tags="car")
```

- After the loop, call $mainloop$ of the window to display on the screen.

```
window.mainloop()
```

- After the $\_\_init\_\_$ function define the function to increase the speed.
    - The function is given a parameter $event$, that will listen for the button pressing event.
    - Increase the speed by increase the amount of dx.

```
def incSpeed(self, event):
    if self.dx < 40:
        self.dx += 5
```

- Define the function to decrease the speed by decreasing the amount of dx.

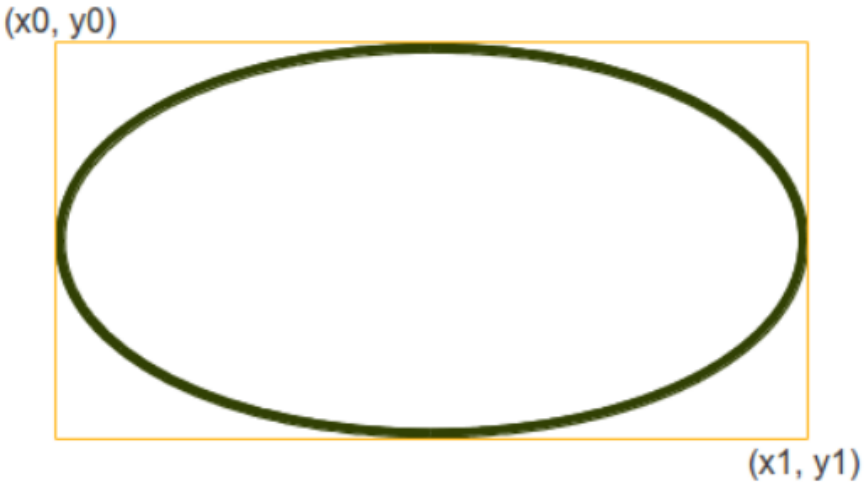
```
def decSpeed(self, event):
    if self.dx > 5:
        self.dx -= 5
```

- Call the class name and run the application.

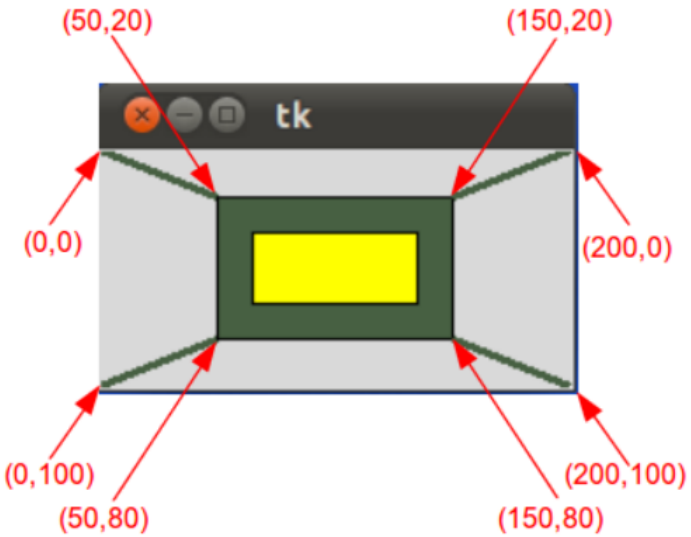
Colored Circle Animation

In this application, we will animate a circle with a random color to fill the window size, and after some time, display an image of explosion with a sound.

In tkinter, there is a function in the Canvas to create ovals. The oval is fit into a rectangle defined by the coordinates (x0, y0) of the top left corner and the coordinates (x1, y1) of a point just outside of the bottom right corner.



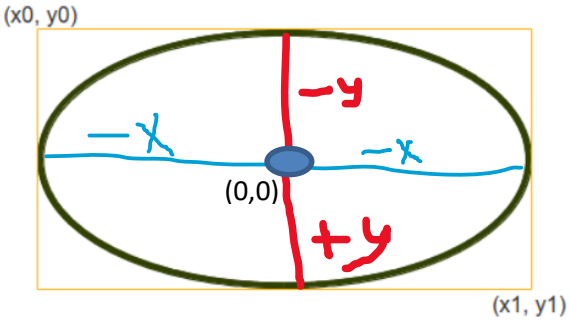
In tkinter, the origin point (0,0) starts at the top-left corner of the window.



To set the origin point to the center of the window, we calculate the center of the window by calculating $cx = \text{WINDOW WIDTH} / 2$ and $cy = WINDOW\ \ HEIGHT / 2$.

To draw a circle around the origin with radius r, we use the following formulas:

- $x_0 = cx - r$
- $y_0 = cy - r$
- $x_1 = cx + r$
- $y_1 = cy + r$



When animating a circle to increase in size, we increase the radius of the cirecle.

Circles.py

```
from random import randrange
from tkinter import *
from winsound import *

# Create the main window
window = Tk()
# Create a canvas for drawing
cnvs = Canvas(window, width=600, height=600)
cnvs.pack()

# Initial parameters for the circles
cx = 300  # Center x-coordinate
cy = 300  # Center y-coordinate
r = 20  # Initial radius
stroke_size = 20  # Initial stroke size
counter = 0  # Counter for the loop

# Loop to create animated circles
while counter < 180:
    # Generate a random fill color
    fill_color = '#%02x%02x%02x' % (randrange(256), randrange(256), randrange(256))

    # Draw the circle on the canvas
    cnvs.create_oval(cx - r, cy - r, cx + r, cy + r, fill=fill_color, width=stroke_size, tags="c")

    # Update the canvas and wait briefly
    cnvs.after(50)
    cnvs.update()

    # Increase the radius and decrease the stroke size
    r += 2
    stroke_size -= 0.09

    # Delete the previous circle
    cnvs.delete("c")

    # Increment the counter
    counter += 1

# Load frames for the GIF animation
frame_count = 20
frames = [PhotoImage(file="exp.gif", format="gif -index %i" % i) for i in range(frame_count)]
```

Circles.py

```
def update(ind):
    # Update the displayed frame for the GIF
    frame = frames[ind]
    ind += 1
    if ind == frame_count:
        ind = 0  # Loop back to the first frame

    # Play sound asynchronously
    PlaySound("exp_snd.wav", SND_ASYNC)

    # Update the label with the new frame
    label.configure(image=frame)

    # Schedule the next update
    window.after(100, update, ind)


# Create a label to display the GIF
label = Label(window)
label.place(x=200, y=200)

# Start the GIF animation
window.after(0, update, 0)

# Run the main loop
window.mainloop()
```

Step-by-step code

- Import the required modules:
  - $randrange$ used to get a random number given a range. This used to get random colors.
  - $winsound$ is a module used to play sounds in the application.

```
from random import randrange
from tkinter import *
from winsound import *
```

- Create a tkinter window and a canvas with width 600 and height 600.

```
window = Tk()
cnvs = Canvas(window, width=600, height=600)
cnvs.pack()
```

- Define:
  - cx, cy, the middle points of the window, which are used as the origin.
  - r, defines the radius of the circle.
  - $stroke\_size$, defines the size of the outer edge of the circle
  - $counter$, count the number of the loops.

```
cx = 300 ; cy = 300 ; r = 20
stroke_size = 20
counter = 0
```

- Define the while loop to run 180 times.
- Inside loop, get a random color, and create an oval filled with this color and the outline width of the circle is 20.

```
while counter < 180:
    fill_color = '#%02x%02x%02x' % (randrange(256), randrange(256),
randrange(256))
    cnvs.create_oval(cx - r, cy - r, cx + r, cy + r,
fill=fill_color, width= stroke_size, tags="c")
```

  - The line '#%02x%02x%02x' means convert the numbers to two-hexadecimal numbers. Colors in tkinter are specified using hex numbers, for example “#40E32F”

- Next, pause the application thread for some time, and update the canvas.
- Generate a random color again.
- Increase the radius of the circle.
- Decresea the stroke size by 0.09.
- Delete the old circle, and redraw it again with the new coordinates.
- Increment the counter by 1.

```
cnvs.after(10)
cnvs.update()
r += 2
stroke_size -= 0.09
cnvs.delete("c")
cnvs.create_oval(cx - r, cy - r, cx + r, cy + r, width=stroke_size,
fill=fill_color, tags="c")
counter += 1
```

- After the animation of circle ends, delete the circle to replace it the gif image of the explosion.
- A gif image does not run automatically on tkinter, instead, we need to explicitly tell tkinter to replace the frames of the image.
    - Suppose that the gif image consistes of 20 frames.
    - We read the image from frame 0 to frame 19.
    - The format='gif -index %i' line means read the image file starting at frame i.

```
cnvs.delete("c")

frame_count = 20
frames = [PhotoImage(file='exp.gif', format='gif -index %i' % (i))
for i in range(frame_count)]
```

- Next, define an update function that will be used to display the sequence of the frames on the window.

- Inside the update function, we will play the sound of the explosion.

```
def update(ind):
    frame = frames[ind]
    ind += 1
    if ind == frame_count:
        ind = 0
    PlaySound("exp_snd.wav", SND_ASYNC)
    label.configure(image=frame)
    window.after(100, update, ind)

label = Label(window)
label.place(x=200, y=200)
window.after(0, update, 0)

window.mainloop()
```

  - The update function takes ind parameter that is used to display the frame at index ind.
  - The line `PlaySound("exp_snd.wav", SND_ASYNC)` is used to play the sound file ASYNCHRONOSLY, meaning that play the file immediately without waiting for another thread.
  - The line `label.configure(image=frame)` displays the image on a label. Images cannot be handedled alone.
  - The line `window.after(0, update, 0)` means call the function update every 0 ms (first zero) and pass the second 0 to the function.

TASK

Create a Tkinter animation application that moves a GIF car image along the x-axis, as shown below. The movement should be slow; aim for the total duration of the animation to be at least 20 seconds. The application should play a music file for the entire duration of the animation. Additionally, the background color of the canvas should change randomly every 1 second.