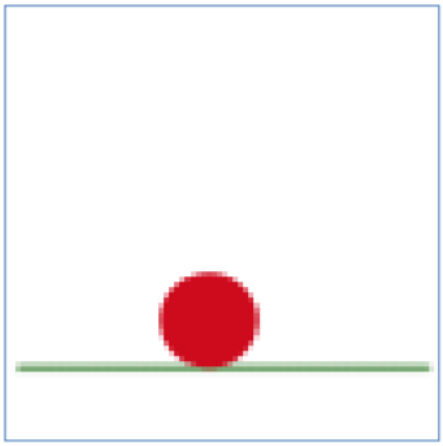
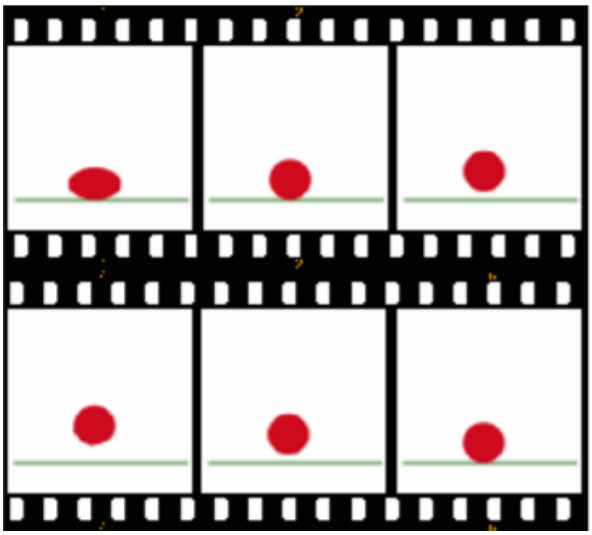
Computer Animation Lab: 01

Contents

Introduction

- Animation is the rapid display of a sequence of images to create an illusion of movement.
- Each frame is a photograph, drawing, or computer-generated image.
- Each frame differs slightly from the one before it.
- Viewing the frames in rapid succession implies “motion”





Animation in Tkinter

- Tkinter is a GUI library used by many programming languages for developing GUI programs on Windows, Mac, and UNIX.
- Tkinter provides an interface for Python to use the Tk GUI library.
- Simple GUI program in Tkinter

SimpleApp

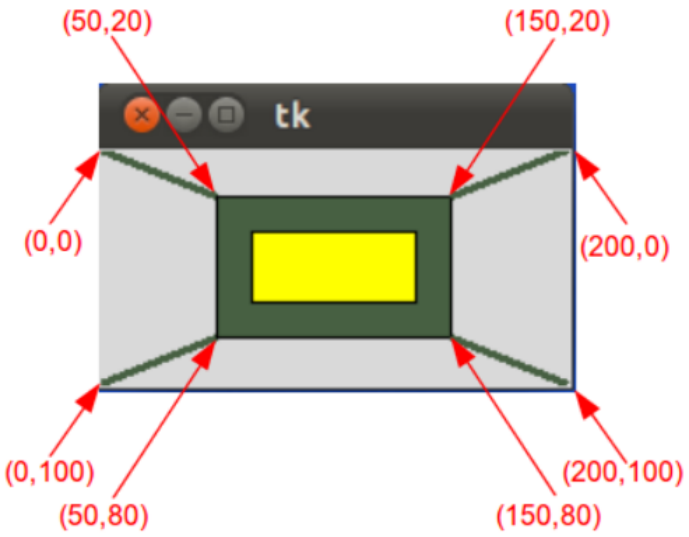
```
from tkinter import *

window = Tk()
lbl = Label(window, text="Welcome to Python")
btn = Button(window, text="Click Me!")
lbl.pack()
btn.pack()

window.mainloop()
```

- $from\ tkinter\ import\ *$
  Import all definitions from tkinter.
- $window\ =\ Tk()$
  Create a window.
- $lbl\ =\ Label(window, text\ =\ "Welcome\ to\ Python")$
  Create a label.
- $btn\ =\ Button(window, text\ =\ "Click\ Me!")$
  Create a button.
- $lbl.pack()$
  Place the label in the window.
- $btn.pack()$
  Place the button in the window.
- $window.mainloop()$
  Create event loop to run the application.

- Animations can be created by displaying a sequence of drawings using $Canvas$ class.
  - The $Canvas$ class can be used to display graphics and develop animations.
- Canvas coordinates in tkinter starts at the top-left corner at (0,0).
  - Go from left to right, increase x-axis.
  - Go from top to down, increase y-axis.



- Use the $move(tags, dx, dy)$ method to move the graphic with the specified tags dx pixels to the right if dx is positive and dy pixels down if dy is positive.
    - If dx or dy is negative, the graphic is moved left or up.
- Animation Demo

AnimationDemo

```
from tkinter import *

window = Tk()
window.title("Animation Demo")

width = 250
cnvs = Canvas(window, bg="white", width = width, height=200)
cnvs.pack()

x = 0
y = 100
cnvs.create_text(x, y, text="Message moving", tags="text")

dx = 3
while True:
    cnvs.move("text", dx, 0)
    cnvs.after(100)
    cnvs.update()
    if x < width:
        x += dx
    else:
        x = 0
        cnvs.delete("text")
        cnvs.create_text(x, y, text= "Message moving", tags="text")

window.mainloop()
```

- $cnvs = Canvas(window, bg = "white", width = width, height = 200)$
  create canvas object on the main window, set background color to white, with the specified width and height.
- $cnvs.pack()$
  display the canvas object on the window.

- $cnvs.create\_text(x, y, text = "Message\ moving", tags = "text")$
  create a text object on the canvas at location x and y. The text object is displayed as "Message moving". The text object is given the identifier "text".
- $cnvs.move("text", dx, 0)$
  move the text object to the right by space dx.
- $cnvs.after(100)$
  Sleep for 100 milliseconds.
- $cnvs.update()$
  update the objects on the canvas to simulate animation.

- We can add tools to control the animation's speed, stop the animation, and resume the animation.

ControlAnimation

```
from tkinter import *

class ControlAnimation:
    def __init__(self):
        window = Tk()
        window.title("Control Animation Demo")
        self.width = 250
        self.height = 50
        self.canvas = Canvas(window, bg="white", width=self.width,
height=self.height)
        self.canvas.pack()

        frame = Frame(window)
        frame.pack()

        bt_stop = Button(frame, text="Stop", command=self.stop)
        bt_stop.pack(side=LEFT)

        bt_resume = Button(frame, text="Resume", command=self.resume)
        bt_resume.pack(side=LEFT)

        bt_faster = Button(frame, text="Faster", command=self.faster)
        bt_faster.pack(side=LEFT)

        bt_slower = Button(frame, text="Slower", command=self.slower)
        bt_slower.pack(side=LEFT)
```

```
        self.x = 0
        self.sleepTime = 100
        self.canvas.create_text(self.x, 30, text="Message Moving?",
tags="text")

        self.dx = 3
        self.isStop = False
        self.animate()

        window.mainloop()

    def stop(self):
        self.isStop = True
    def resume(self):
        self.isStop = False
        self.animate()
    def faster(self):
        if self.sleepTime > 5:
            self.sleepTime -= 20
    def slower(self):
        self.sleepTime += 20
    def animate(self):
        while not self.isStop:
            self.canvas.move("text", self.dx, 0)
            self.canvas.after(self.sleepTime)
            self.canvas.update()
            if self.x < self.width:
                self.x += self.dx
            else:
                self.x = 0
                self.canvas.delete("text")
                self.canvas.create_text(self.x, 30, text="Message
Moving?", tags="text")

ControlAnimation()
```

- *class* $ControlAnimation$:
  This line defines a class $ControlAnimation$ that describes the behavior of the whole application.

- The $ControlAnimation$ class has 6 functions

```
class ControlAnimation:
    def __init__(self):...
    def stop(self):...
    def resume(self):...
    def faster(self):...
    def slower(self):...
    def animate(self):...
```

  - The $\_\_init\_\_(self)$ is a special function to initialize object's attributes (similar to a constructor in Java and C++).
  - The $self$ parameter represents an instance of the class. This instance will hold the attributes and functions of the class – that is why is passed in every function.
  - The `stop(self)` function defines the events that will be executed when press $Stop$ button. In this case, it stops the animation by setting $isStop$ variable to $True$.
  - The `resume(self)` function defines the event when pressing the $Resume$ button. In this case, it resumes the animation by setting $isStop$ to $False$.
  - The `faster(self)` function defines the event when pressing the $Faster$ button. In this case, it increases the animation's speed by decreasing the sleep time of the $Canvas$.
  - The `slower(self)` function defines the event when pressing the $Slower$ button. In this case, it decreases the animation's speed by increasing the sleep time of the $Canvas$.
  - The `animate(self)` function defines the animation process.
- In the $\_\_init\_\_$ function
  - window = Tk()
    create a new tkinter window
  - window.title("Control Animation Demo")
    set the title of the window

- `self.width=250` and `self.height=50`
  sets the width and height of the window
- self. canvas = Canvas(window, bg = "white", width = self. width, height = self. height)
  creates a Canvas object that will host the animating text. The *window* option means that the canvas is set on the window object we created. $bg = white$ means that the canvas has a white background.
- $pack()$ is a function that displays the object on the window.
- frame = Frame(window)
  create a frame object with its parent the *window*. This used to handle the buttons.
- $bt\_stop = Button(frame, text = "Stop", command = self.stop)$
  create a button with its parent $frame$. The $text$ option defines the label appears on the button. The $command$ option define what function to invoke when pressing on the button. The same applies for the bt_resume , bt_slower and `bt_faster`.
- `bt_stop.pack(side=LEFT)`
  set the button on left side on the $frame$. The same applies for the bt_resume , bt_slower and bt_faster.
- self. x = 0
  the variable x defines the x-axis location of the text object.
- `self.sleepTime=100`
  The $sleepTime$ variable defines the time in millisecond to pause the motion of the text.
- $self.canvas.create\_text(self.x, 30, text = "Message Moving?", tags = "text")$
  create a text object on the canvas at location (x, 30). The object has its label text “message moving?”. The object has the identifier “text”.
- `self.dx=3`
  The dx variable defines the amount of x-axis distance to move.

- `self.isStop=False`
  A Boolean variable to stop or resume the animation.

- In the $animate()$ function
  - `while not self.isStop:`
    Set up a $while$ loop that runs as long as the $isStop$ variable is $False$.
  - `self.canvas.move("text", self.dx, 0)`
    Move the text object a distance dx along x-axis and a distance 0 along y-axis.
  - `self.canvas.after(self.sleepTime)`
    Pause the animation thread for $sleepTime$ millisecond.
  - `self.canvas.update()`
    Refresh the canvas content after processing all events, i.e., update the position of the text variable after recalculating its x-axis position
  - ```
    if self.x < self.width:
        self.x += self.dx
    ```
    If the position of the text has not reached the end of the window, move it to the right.
  - ```
    else:
        self.x = 0
        self.canvas.delete("text")
    self.canvas.create_text(self.x, 30
    ,text="Message Moving?", tags="text")
    ```
    Otherwise, if the text object has reached the end of the window, set its x-axis location to 0 (the beginning of the screen), and delete the text object, then recreate it with the new x-axis position.

Checkpoint

- Try setting $dx = 15$, what do you notice?
- How to change the canvas background color to black and change text color to white.

Bouncing Ball

BouncingBall

```
from tkinter import *

window = Tk()
window.title("BouncingBall")
width = 400
height = 300

cnvs = Canvas(window, bg="white", width=width, height=height)
cnvs.pack()
# Create a circle
circle_radius = 20
x = circle_radius
y = height // 2

dx = 2
dy = 2
# Draw the circle
circle = cnvs.create_oval(x - circle_radius, y - circle_radius,
                          x + circle_radius, y + circle_radius, fill="blue")

def animate():
    global x, dx, y, dy
    # Move the circle
    cnvs.move(circle, dx, dy)
    x += dx
    y += dy
    # Bounce off the edges
    if x + circle_radius >= width or x - circle_radius <= 0:
        dx = -dx  # Reverse direction
    # Bounce off the edges
    if y + circle_radius >= height or y - circle_radius <= 0:
        dy = -dy  # Reverse direction

    # Schedule the next frame
    window.after(20, animate)


# Start the animation
animate()

# Start the Tkinter event loop
window.mainloop()
```

Introduction to Blender

- With Blender, you can create 3D visualizations such as still images, 3D animations, VFX shots, and video editing.
- Features
    - Blender is a fully integrated 3D content creation suite, offering a broad range of essential tools, including Modeling, Rendering, Animation & Rigging, Video Editing, VFX, Compositing, Texturing, and many types of Simulations.
    - It is cross platform, with an OpenGL GUI that is uniform on all major platforms (and customizable with Python scripts).
    - It has a high-quality 3D architecture, enabling fast and efficient creation workflow.
    - It boasts active community support, see blender.org/community for an extensive list of sites.
    - It has a small executable, which is optionally portable.
- Download Blender 4.2 from https://www.blender.org/download/

Exercise

- How to make the text move along the y-axis only?