Problem Solving through Programming (COM404)

19 - Animation

Problem Solving through Programming

Activity 1: Animating a Single Component (~ minutes)

[Explanation]

An animation is simply a series of images displayed in quick succession. In order to add animation to our graphical user interface we will need to use a timer that can be used to synchronise our animated components. We can do this by importing the time module as shown in the example below:

```
from tkinter import *
import time

# the class
class AnimatedGui(Tk):
    def __init__(self):
        super().__init__()

        # load resources

        # set window attributes

        # add components

# the object
if __name__ == "__main__":
    gui = AnimatedGui()
    gui.mainloop()
```

We can then create our timer function which will run every so often and which will control our animation. To do this we will use TkInter's after function. For example, we can call our timer function tick and make it run every 100 milliseconds:

```
from tkinter import *
import time

# the class
class AnimatedGui(Tk):
    def __init__(self):
        super().__init__()

        # load resources

        # set window attributes

        # add components

        # start the timer
        self.tick()


    # the timer tick function
    def tick(self):
        self.after(100, self.tick)

# the object
if __name__ == "__main__":
    gui = AnimatedGui()
    gui.mainloop()
```

Now that we have setup a basic timer to control our animation we can start animating our components. As an example, we will create a basic animation of a bouncing ball. First of all we need to set the dimension of our window and load in the image of our ball. As we will be moving the ball around our window, we will use the place layout manager.

```
from tkinter import *
import time

# the class
class AnimatedGui(Tk):
    def __init__(self):
        super().__init__()

        # load resources
        self.ball_image = PhotoImage(file="ball.gif")

        # set window attributes
        self.configure(height=500,
                       width=500)

        # add components
        self.add_ball_image_label()

        # start the timer
        self.tick()

    # the timer tick function
    def tick(self):
        self.after(100, self.tick)

    # the ball image
    def add_ball_image_label(self):
        self.ball_image_label = Label()
        self.ball_image_label.place(x=200, y=200)
        self.ball_image_label.configure(image=self.ball_image)

# the object
if __name__ == "__main__":
    gui = AnimatedGui()
    gui.mainloop()
```

In order to move the ball we will need to track its x and y position. We can do this by introducing two variables to remember the current x and y position of the ball.

```
from tkinter import *
import time

# the class
class AnimatedGui(Tk):
    def __init__(self):
        super().__init__()

        # load resources
        self.ball_image = PhotoImage(file="ball.gif")

        # set window attributes
        self.configure(height=500,
                       width=500)

        # set animation attributes
        self.ball_x_pos = 200
        self.ball_y_pos = 200

        # add components
        self.add_ball_image_label()

        # start the timer
        self.tick()

    # the timer tick function
    def tick(self):
        self.after(100, self.tick)

    # the ball image
    def add_ball_image_label(self):
        self.ball_image_label = Label()
        self.ball_image_label.place(x=self.ball_x_pos,
                                    y=self.ball_y_pos)
              self.ball_image_label.configure(image=self.ball_image)

# the object
if __name__ == "__main__":
    gui = AnimatedGui()
    gui.mainloop()
```

To move the ball we can update the x and y position of the ball on each 'tick' of the timer. We do this by updating the value of x and y in the tick method:

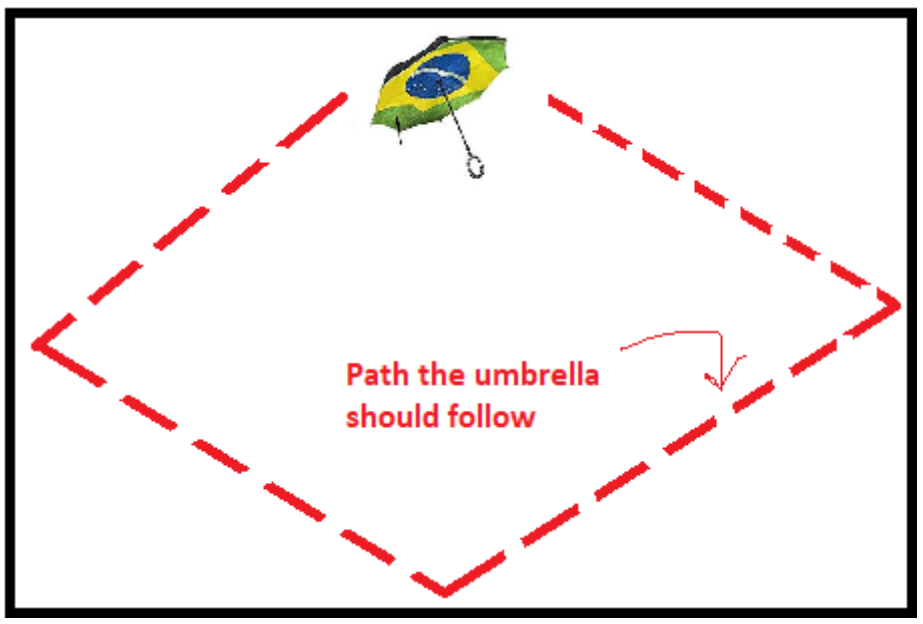
```
from tkinter import *
import time

# the class
class AnimatedGui(Tk):
    def __init__(self):
        super().__init__()

        # load resources
        self.ball_image = PhotoImage(file="ball.gif")

        # set window attributes
        self.configure(height=500,
                       width=500)

        # set animation attributes
        self.ball_x_pos = 200
        self.ball_y_pos = 200

        # add components
        self.add_ball_image_label()

        # start the timer
        self.tick()

    # the timer tick function
    def tick(self):
        self.ball_x_pos = self.ball_x_pos + 1
        self.ball_y_pos = self.ball_y_pos + 1
        self.ball_image_label.place(x=self.ball_x_pos,
                                    y=self.ball_y_pos)
        self.after(100, self.tick)

    # the ball image
    def add_ball_image_label(self):
        self.ball_image_label = Label()
        self.ball_image_label.place(x=self.ball_x_pos,
                                    y=self.ball_y_pos)
              self.ball_image_label.configure(image=self.ball_image)

# the object
if __name__ == "__main__":
    gui = AnimatedGui()
    gui.mainloop()
```

If we wish to be able to change the direction or the amount by which the ball moves then we can introduce two additional variables to track the change in the x position and the change in the y position as shown below:

```
from tkinter import *
import time

# the class
class AnimatedGui(Tk):
    def __init__(self):
        super().__init__()

        # load resources
        self.ball_image = PhotoImage(file="ball.gif")

        # set window attributes
        self.configure(height=500,
                       width=500)

        # set animation attributes
        self.ball_x_pos = 200
        self.ball_y_pos = 200
        self.ball_x_change = 1
        self.ball_y_change = 1

        # add components
        self.add_ball_image_label()

        # start the timer
        self.tick()

    # the timer tick function
    def tick(self):
        self.ball_x_pos = self.ball_x_pos + self.ball_x_change
        self.ball_y_pos = self.ball_y_pos + self.ball_y_change
        self.ball_image_label.place(x=self.ball_x_pos,
                                    y=self.ball_y_pos)
        self.after(100, self.tick)

    # the ball image
    def add_ball_image_label(self):
        self.ball_image_label = Label()
        self.ball_image_label.place(x=self.ball_x_pos,
                                    y=self.ball_y_pos)
        self.ball_image_label.configure(image=self.ball_image)

# the object
if __name__ == "__main__":
    gui = AnimatedGui()
    gui.mainloop()
```

[Tasks]

Create a program with an animation of a component bouncing around the window as shown in the image below. You can use any GIF image or even create your own using a suitable drawing package.



[Commit to Repository]

It is time to commit our working program to our GitHub repository. Commit each of your files to the following location:

```
2-guis/5-animation/1-single-object/
```

Once you are ready to commit each file to the repository, add a description for the commit summarising the changes you have made e.g. "Added code to animate a component bouncing in the window." and then click the commit button.

Problem Solving through Programming

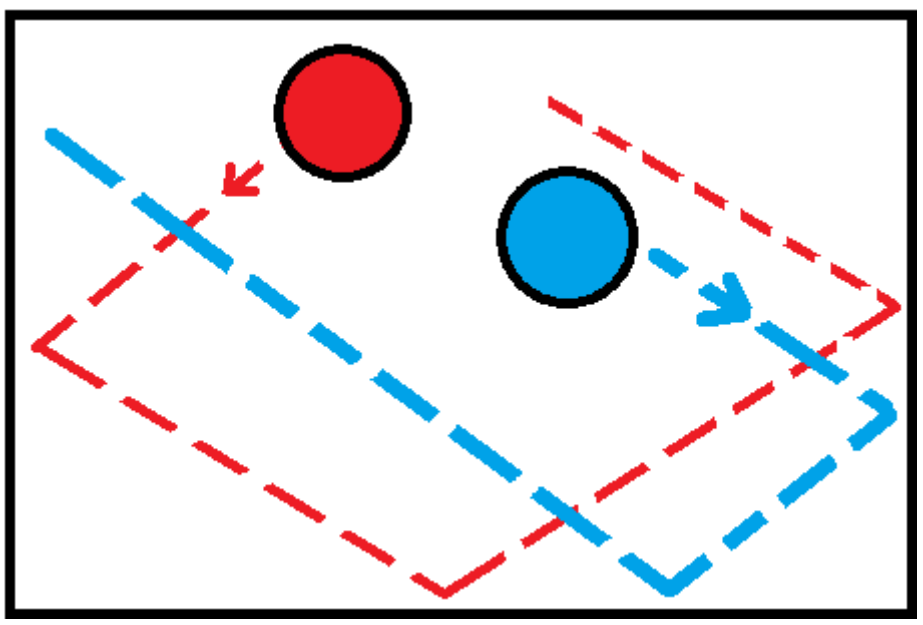
Activity 2: Animating Multiple Components(~ minutes)

[Explanation]

It is possible for us to animate multiple components in our timer's tick function or indeed have multiple timers. In order animate multiple components we simply add any required attributes to our __init__ function and animate the components in our timer's tick function. For example, if we have two balls - a red ball and a blue ball - the we wish to animate in our tick function, then this would look similar to the following:

[Tasks]

Create a program with an animated component bouncing around the window as shown in the image below. You can use any GIF image or even create your own using a suitable drawing package.



[Commit to Repository]

It is time to commit our working program to our GitHub repository. Commit each of your files to the following location:

```
2-guis/5-animation/2-multiple-components/
```

Once you are ready to commit each file to the repository, add a description for the commit summarising the changes you have made e.g. "Added code to animate multiple components in the window." and then click the commit button.

Problem Solving through Programming

Activity 3: Animating Multiple Components at Multiple Speeds(~ minutes)

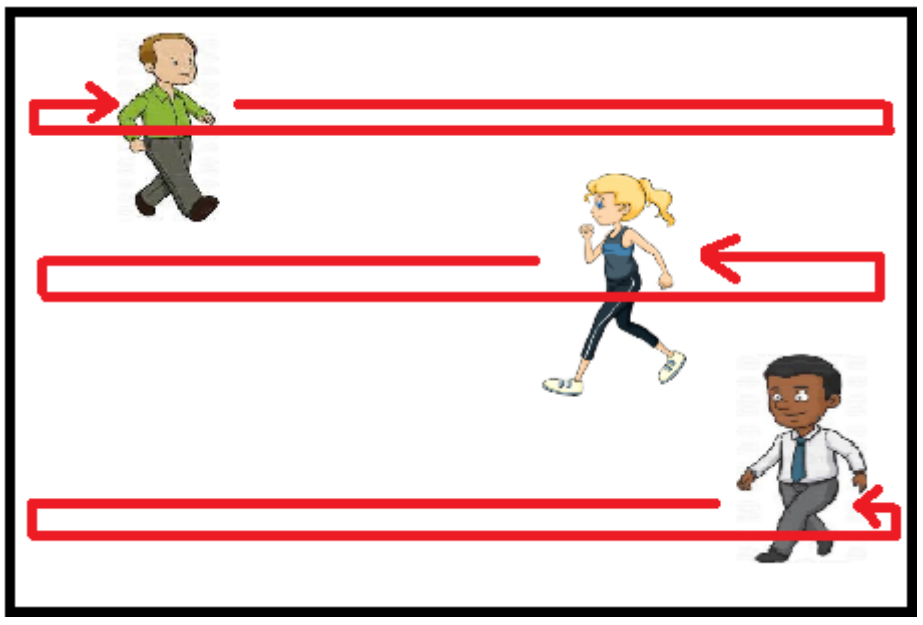
[Explanation]

It is sometimes desirable to animate a component after several ticks rather on each tick of the timer. We can use the modulus operator (%) with a suitable variable to control the number of ticks after which a component should be animated. For example, let us say in the previous example we wanted the blue ball to move on every 4th tick. We would need a variable to track the number of ticks and we would need to use the % operator as follows:

```
...
    if (self.num_ticks % 4 == 0):
        ...
...
```

[Tasks]

Create a program that consists of an animation showing people moving around as illustrated by the figure below. Each person in the animation should move after a different number of ticks and at different speeds. You can use any GIF images or even create your own using a suitable drawing package.



[Commit to Repository]

It is time to commit our working program to our GitHub repository. Commit each of your files to the following location:

```
2-guis/5-animation/3-varying-ticks/
```

Once you are ready to commit each file to the repository, add a description for the commit summarising the changes you have made e.g. "Added code to animate moving people." and then click the commit button.

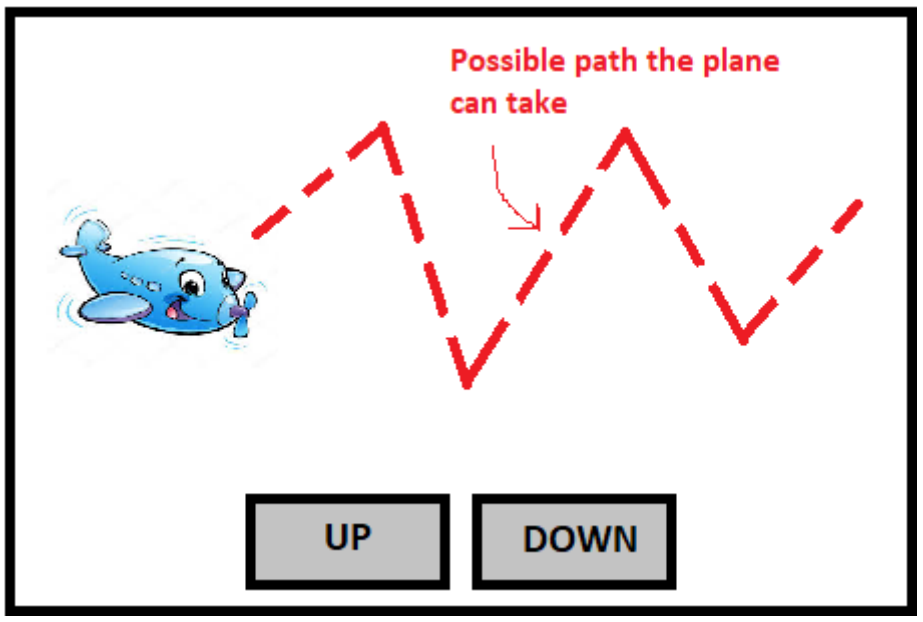
Activity 4: Animating with Events(~ minutes)

[Explanation]

Once we have defined attributes to control our animation, we can also manipulate these attributes when certain events occur. For example, we can change the horizontal or vertical direction of a moving component by reversing its the value of the attributes controlling its x or y position respectively.

[Tasks]

Create a program that consists of an animation where a plane (or some other image) flies from side to side and moves up/down when the user clicks the up/down buttons as shown in the image below:



Implement the program described above. You can use any appropriate GIF images or even create your own using a suitable drawing package.

[Commit to Repository]

It is time to commit our working program to our GitHub repository. Commit each of your files to the following location:

```
2-guis/5-animation/4-with-events/
```

Once you are ready to commit each file to the repository, add a description for the commit summarising the changes you have made e.g. "Added code to animate moving people." and then click the commit button.

Last modified: Tuesday, 26 November 2019, 4:59 AM

◄ 18 - Images

Jump to...

AE2 - Review TCA 1 ►