DogBark – Iterative Enhancement Plan

Do this entire project ***live in-class*** with your instructor.
(Or use the associated videos, although they are out of date.)

Test your program (by running it) after EACH of the following **bold** steps.

1. Make an "empty" project (that does nothing), as follows:

a. Add your name as an author of the program.

b. Add the necessary imports (just *pygame* and *sys* for this program).

c. Define *global constants* that we will want for:

`WHITE = pygame.Color("white")  # or (255, 255, 255)`

IMAGE_SIZE = 470 TEXT_HEIGHT = 30

Note: Use ALL_CAPS for these names, as always for constants.

d. Define a *main* function and call it at the bottom of the file.

e. Define empty "sections" of *main* for:

- Initialization
- The game loop, that is: `while True:`
- Inside the game loop: *Respond to events* and *Draw things*.

2. Display (only) a white screen (window), as follows:

a. In the *Initialization* section of main, initialize pygame.

b. Also make a screen, by using the ***pygame.display*** package to set the *mode* to size:

`(IMAGE_SIZE, IMAGE_SIZE + TEXT_HEIGHT)`

c. Also make a *caption* for the game window:
"Text, Sound, and an Image".

d. In the *Game Loop*, in the *Respond to Events* section, get and store the events by using the ***pygame.event*** package.

e. In the *Game Loop*, in the *Drawing* section, fill the screen with WHITE by using the ***screen's fill*** method, and *update* the display by using the ***pygame.display*** package.

3. Respond to the `pygame.QUIT` event by `sys.exit()`

4. Display an image, by:

a. In the Initialization, use the ***pygame.image*** package to load an image from the *2dogs.jpg* file:

```
dog_image = pygame.image.load("2dogs.jpg")
```

b. In the Drawing section of the game loop , use the ***screen's blit*** method to display the image onto the screen at (0, 0):

```
screen.blit(dog_image, (0, 0))
```

5. *Scale* the image to be the size `(IMAGE_SIZE, IMAGE_SIZE)`, by using the ***pygame.transform*** package like this, just after *loading* the image:

```
dog_image = pygame.transform.scale(
                dog_image, (IMAGE_SIZE, IMAGE_SIZE))
```

6. (Temporary step) Find the fonts that are on your computer.

Use the ***pygame.font*** package to *get* the fonts that are on your computer, then print the resulting list in a loop. Also *get* and print the default font for your computer.

7. Display the text "Two Dogs" on the image.

a. In the Initialization section, use the ***SysFont*** method of the ***pygame.font*** package to create a Font object (called, say, *font1*) in a font of your own choosing, at size 40 or so.

b. Then load the string "Two Dogs" into a caption, like this:

```
caption1 = font1.render("Two Dogs", True, (255, 0, 255))
```

Since this caption will not change during the program run, put the above statement into the Initialization section. The argument *True* means to *use antialiasing* (smoothing) and the last argument is the *color* to use (here, a neon pink).

c. Draw the caption in the Drawing section by using the ***screen's blit*** method, just like you did for the image. Put it wherever you want on the screen.

[continues on the back of this page]

8. **Display some other text (keep it "G" rated!) on the image, using a different font and font size, centered in the white space at the bottom of the screen.** The screen and caption's *get_width* methods are useful for horizontal centering. Use the screen's *get_height* method and the TEXT_HEIGHT constant for the vertical placement in the white space at the bottom of the screen.

9. **Play the "bark.wav" file when the mouse is clicked.**

   a. In the Initialization section, use the ***pygame.mixer*** package to make a ***Sound*** object from the *bark.wav* file.

   b. In the Respond to Events section, when an event whose type is pygame.MOUSEBUTTONDOWN occurs, *play* the Sound object.

10. **Stream the "whip-110235.mp3" file when the program starts.**

```
pygame.mixer.music.load("whip-110235.mp3")
pygame.mixer.music.play()
```