**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(Or use the associated videos, although they are out of date.)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:**
   1. Add your name as an author of the program.
   2. Add the necessary imports (just *pygame* for this program).
   3. 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.

* 1. Define a *main* function and call it at the bottom of the file.
  2. Define empty “sections” of *main* for:
     + Initialization
     + The game loop, that is: while True:
     + Inside the game loop: *Respond to events* and *Draw things*.

1. **Display (only) a white screen (window), as follows:**
   1. In the *Initialization* section of main, initialize pygame.
   2. Also make a screen, by using the ***pygame.display*** package to set the *mode* to size:

(IMAGE\_SIZE, IMAGE\_SIZE + TEXT\_HEIGHT)

* 1. Also make a *caption* for the game window:  
      *"Text, Sound, and an Image".*
  2. In the *Game Loop*, in the *Respond to Events* section, get and store the events by using the ***pygame.event*** package.
  3. 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.

1. **Respond to the pygame.QUIT event by sys.exit()**
2. **Display an image, by:**
   1. In the Initialization, use the ***pygame.image*** package to load an image from the *2dogs.jpg* file:

dog\_image = pygame.image.load("2dogs.jpg")

* 1. 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))

1. ***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))

1. **(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.

1. **Display the text "Two Dogs" on the image.**
   1. 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.
   2. 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).

* 1. 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]

1. **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.
2. **Play the “bark.wav” file when the mouse is clicked.**
   1. In the Initialization section, use the ***pygame.mixer*** package to make a ***Sound*** object from the *bark.wav* file.
   2. In the Respond to Events section, when an event whose type is pygame.MOUSEBUTTONDOWN occurs, *play* the Sound object.
3. **Stream the “whip-110235.mp3” file when the program starts.**

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

pygame.mixer.music.play()