**Python Lab 10a Images in Code Skulptor**

**For this chapter, we will work in the older version of Code Skulptor, which uses Python 2**

|  |  |
| --- | --- |
| 0. Go to CodeSkulptor.org In the documentation, search for “image.”  Copy the sample code for simplegui.loadimage() into your program……...a copy of code is given below:  **import simplegui  def draw\_handler(canvas):  canvas.draw\_image(image, (1521 / 2, 1818 / 2), (1521, 1818), (50, 50), (100, 100))  image = simplegui.load\_image('.................')  frame = simplegui.create\_frame('Testing', 100, 100) frame.set\_draw\_handler(draw\_handler) frame.start()**  Pretend you want to load a different image with a width and height of (400, 300).  You want to draw it in a frame that has a width of 150 by 100. Write the code, using this as a hint:  **canvas.draw\_image(image, center\_source, width\_height\_source, center\_dest, width\_height\_dest)**  **canvas.draw\_image(image,\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Check your work before you continue.** | |
| 1. Add the following line (you decide where):  **rotation = 0**  Modify existing code to match these lines: (you decide where it goes)  **canvas.draw\_image(image, (2020/2, 1866/2), (2020, 1866), (150, 150),(300, 300), rotation)**  **image = simplegui.load\_image('https://tinyurl.com/y7jee9ae')**  **frame= simplegui.create\_frame(‘Image Test’, 300, 300)** | |
| 2. Add the following lines:  **delta\_r = 0**  **def keydown(key):**  **global delta\_r**  **if key == simplegui.KEY\_MAP['left']:**  **delta\_r = -0.01**  **elif key == simplegui.KEY\_MAP['right']:**  **delta\_r = 0.01**  Add these lines to the draw handler:  **global rotation**  **rotation = rotation + delta\_r**  **print rotation**  Add this line near the bottom:  **frame.set\_keydown\_handler(keydown)** | 3. Use a keyup handler to stop the rotation when the key is let go:  **def keyup(key):**  **global delta\_r**  **…………...**  **frame.set\_keyup\_handler(keyup)** |
| 4. Pick a different image of your choice. Figure out its width and height. Load it and draw it. |
| 5. Challenge. Find a way to use only the top left quadrant of the image. Make it show up in the entire canvas frame. |
| 6. Challenge. Use the full image, but scale it down to occupy a small part of your frame. Move it around with WASD keys. |

**Python Lab 10b Implementing Timers**

|  |  |
| --- | --- |
| 0. Look up how to make a timer handler. | At the top, make a comment for your name.  In the Docs, search for **timer**. Write the general syntax for making a timer.  Create a timer that runs once every second.  Inside the timer function, print a positive message to the console. |
| 1. Implement a counter that increments once every second. | Instead of printing a message, have the timer print the value of a counter that goes up by 1. Print the counter’s value to the console. Modify your draw handler so that the counter draws in the upper right corner of the screen. |
| 2. Make a toggle button for the timer. | In the Docs, read all the timer commands. List three additional ones below:  Make a button that toggles the timer. When you click the button, the timer stops. The timer is off; then, the timer starts when you click the button.  You may not use any extra variables, use the commands you wrote above. |
| 3. Draw a circle. Save your work. | Add code to the draw handler that draws a circle.  Make global variables for the circle’s x, y, radius, and color.  Use these global variables when drawing the circle.  Save your program. |
| 4. Make the random circle change every half second. | In the timer handler, change the four global variables of the circle to four random values.  The center of your circle could be anywhere in your frame, the radius could be any integer between 1 and 100, and the color should be one of four random colors. Change the timer to run every half of a second. You should see a random circle appear every half of a second. |
| 5. Make the entire circle stay on the screen. | Write code in the timer handler. If the radius is so large that it places any part of the circle off the canvas, then change the radius to be a number is the largest possible radius that keeps the entire circle on the screen. |

**Python Lab 10c Sprite Sheets**

|  |  |
| --- | --- |
| 0. Load the image and print out its size. Do not try to draw the image yet. | In the Code Skulptor documentation, search for “image.”  Write the commands to load an image, get its width, and get its height.  **load the image** [**http://tinyurl.com/zf9ctqa**](http://tinyurl.com/zf9ctqa)  image =    width = image.get\_width()  height =  print (width, height)  x = width/2 # x coordinate of the center of what we are drawing  y = height/2 # y coordinate of the center of what we are drawing |
| 1. Draw the entire image. | In the draw handler, remove the draw\_text line and add these lines:  **source\_dim = (width, height)**  **source\_center = (x , y )**  **dest\_dim = ( \_\_\_\_\_\_\_\_\_\_ , \_\_\_\_\_\_\_\_\_\_\_) #size of your frame**  **dest\_center = (\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_) # center of your frame**  **canvas.draw\_image(image, source\_center, source\_dim, dest\_center, dest\_dim)** |
| 2. Sketch out the sprite sheet. | Now that you see the image, notice that the image is made up of several images of the same sprite. This is called a “sprite sheet.”  On the back of this paper, draw a large grid in which each rectangle in your grid represents each individual image on this sheet. Your teacher will draw a sample grid for the class to see. Write the dimensions of the whole image on your drawing.  Then, draw a dot in the center of each grid. Write the coordinates of each dot. |
| 3. Make a button to set parameters to show one picture. | Rename your button “image1.” It will change the value of the global variables to show only the first sprite on this sheet.  **global x, y, width, height**  **x = \_\_\_\_ #x coordinate of the center of the 1st image**  **y = \_\_\_\_ #y coordinate of the center of the 1st image**  **width = \_\_\_ #width of this sprite**  **height = \_\_\_\_ #height of this sprite** |
| 4. Make more buttons. | Make five more buttons that will change the value of the global variables to show an additional five sprites of this sheet. Name the buttons image2, image3, and so on. |
| 5. Redo this process for a different sprite sheet. | Find a different sprite sheet online. Repeat this entire process for that sprite sheet. No matter what size the sprite sheet is, you will still have six buttons. |

**Python Lab 10d Animating Sprite Sheets**

|  |  |
| --- | --- |
| 0. Load a new image. See it on your canvas. | Reuse the code from the previous assignment:  **image = simplegui.load\_image(‘....’)** [**http://www.xojo3d.com/images/sprite1.png**](http://www.xojo3d.com/images/sprite1.png)  **print (image.get\_width(), image.get\_height())**  Draw the entire image on your canvas.  Remove the buttons and button handlers. |
| 1. Draw the first image. | At the top of your program, create the following variables and give them values:  **num\_rows = \_\_ # number of rows in your sprite sheet**  **num\_cols = \_\_ # number of columns in your sprite sheet**  **sprite\_width = image.get\_width() / num\_cols**  **sprite\_height = image.get\_height()/ num\_rows**  **x = sprite\_width / 2**  **y = sprite\_height / 2** |
| 2. Sketch out the sprite sheet. Then, make a timer and a timer handler. | On the back of this paper, draw a large grid in which each rectangle in your grid represents each individual image on this sprite sheet. Include the width and height. Then, draw a dot in the center of each grid. Write the coordinates of the first row of dots.  Make a timer that is called every half second, and a timer handler function. Inside the timer handler function, print out the values of x and y:  **print (x, y)** |
| 3. Scroll through rows 1 and 2. | At the top of your timer handler, add these lines:  **global x**  **x = x + width**  Run your program. It should change the image drawn, then go off the screen.  Then, add code that does this:  If x gets too large,  x = # initial x value (recalculated, not actual number)  y = # the y value of the next row (calculated, not actual number) |
| 4. Scroll through all rows and back up to the first row. | Figure out how to continuously scroll through all images. After you show the last sprite, go back to showing the first. |
| 5. Make this work for a different image. | Find a different sprite sheet, that we have not used before.  Generalize your code. Ask the user to enter the URL, the number of rows, the number of columns, and the program will run with no changes. |