Ancient Text Fixer

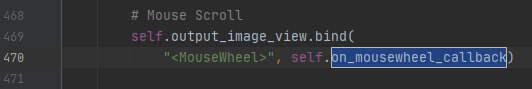
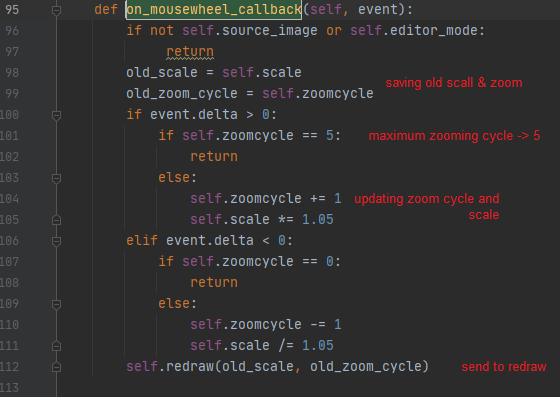
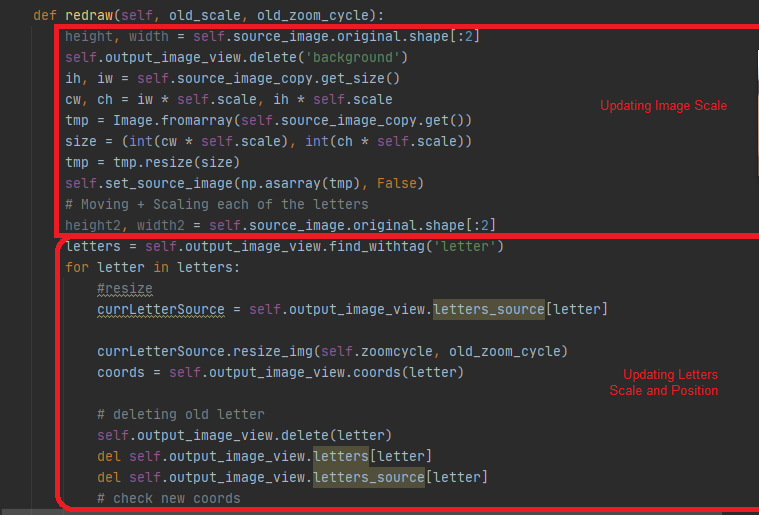
FEATURE General DOCUMENTION

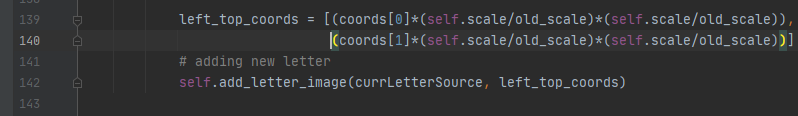
ניתאי סולומון

‏2020

Zoom:

The zoom is built so it will preserve the images as many interpolations may cause a disorder in the images visibility (both of the background and the letters). Each time you add a letter, it adds its source code and source zoom cycle -> when zooming in/out the source code is used with the delta of |source zoom cycle – current zoom cycle| in order to make the cleanest interpolation possible.  
The steps of building the Zoom are described below:

1. Binding mouse wheel to zoom callback -> func: get\_output\_image\_canvas(main:469)  
   
2. Setting the zoom cycle possible(Now its 5) and updating the scale if needed ->  
   func: on\_mousewheel\_callback(main:95)  
   
3. Redraw background and all letters using the new scale and position calculated->  
   func: redraw(main:114)  
   



Edit\_Mode:

The edit mode is a way of running interpolation on a selected image.  
In order to enter Edit-Mode press 'R' then u'll be given a choice to choose your interpolation("nearest b-linear cubic"), Deformat -> using the chosen interpolation, Save or Close discarding all changes made.  
The edit Mode is a pop up invoked by calling the function "popup\_editor()"(main:563).  
By entering this mode it cuts the letter background taking into account the scale and the position of the letter.

All function regarding the options of "edit\_mode" is defined there

Deformator:

The main function of the Deformator class is in charge of making the interpolation of the image. When pressing 'R' and Entering the edit mode of a selected letter the deformat option will call this function with all relevant details, ie: the image source(nparray), 4 points, and the interpolation-0;nearest,1;b-linear;2;cubic.

def deformat\_img(img, x1, y1, x2, y2, interpolation):  
 global img2, vecSecondToFirst  
 interpolations = [nearest\_neighbor, b\_linear, cubic]  
 img2 = img.copy()  
 radius = calculate\_distance(x2, y2, x1, y1)  
 img\_result = img.copy()  
 first\_circle = set() #Second Circle pixels list  
 second\_circle = set() #First Circle pixels list  
 vecSecondToFirst = np.array([x2 - x1, y2 - y1])  
 for b in range(img.shape[0]):  
 for a in range(img.shape[1]):  
 try:  
 if calculate\_distance(x2, y2, a, b) < radius:  
 first\_circle.add((b, a))  
 elif calculate\_distance(x1, y1, a, b) < radius:  
 second\_circle.add((b, a))  
 except:  
 continue  
 far\_shape = second\_circle.difference(first\_circle)  
 for (b, a) in far\_shape:  
 try:  
 x\_res, y\_res = compress\_for\_point(a, b, radius, x1, y1)  
 img\_result[b][a] = interpolations[interpolation](x\_res, y\_res)  
 except:  
 continue  
 for (b, a) in first\_circle:  
 try:  
 x\_res, y\_res = compress\_for\_point(a, b, calculate\_distance(x1, y1, x2, y2) + radius, x1,y1)  
 img\_result[b][a] = interpolations[interpolation](x\_res, y\_res)  
 except:  
 continue  
 return img\_result

The Deformator class contains a function for each interpolation being used by the user.