Natural Waters Defense Foundation

Designers:
Loic Scomparin &
Sragvi Vadali

CSP per 5

C9 Workspace

Table of Contents

Brainstorming Notes & Sketch of Proposed Idea

Client 1:

Your client is a group that advocates for a political cause. It could be the environment, education, anything. The client needs a consistent branding for images that will be used to promote their cause – images that are memorable and will have a lasting impact on people. The client could be a real or fictitious student organization, community group, or state/ national/global advocacy group. The client's cause should be a true potential cause even if the client is fictitious.

The client wants an automated process to apply to images. They want the process to use some combination of masking, shading, or combining the images with a consistent logo or superimposed image. The client enjoys abstract art as well and might like geometric shapes incorporated in the image – drawn on, as a border, or as a mask. The client enjoys participating in the creative process and will appreciate being offered a range of options (as a parameter) for one of the image operations you perform.

We chose project Client 1 - a Cause- and brainstormed possible causes that we could represent:

Brainstorm:	Cause # 1
education	politics
humanitarianism	philanthropy
economy	environment

We chose environment as our topic, and upon further discussion, chose pollution in natural bodies of water as our specific topic.

We decided that the main goals our client organization wanted to achieve were promote donations to their site to help fund education and cleanup efforts, and also urge people to produce less waste, recycle, etc.

We then defined our client, the fictitious national organization Natural Waters Defense Foundation.

Finally, we summarized all that we had done so far into a design brief:

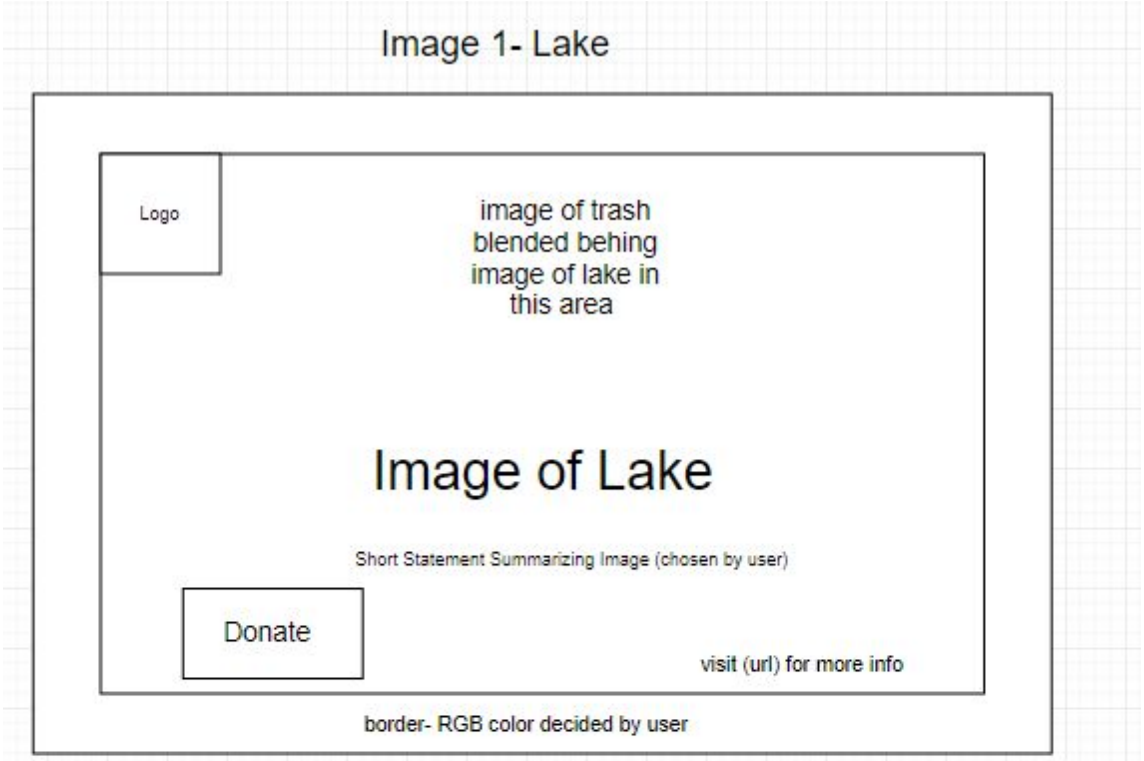
Client	Natural Waters Defense Foundation
Client Objective	Advocate for proper disposal of trash/no dumping in oceans and environmental cleanup of trash that is already there.
Problem Statement	- illegal and legal dumping of trash and chemicals in waterways and surrounding land - create and manipulate images to effectively show the issue and prompt people to donate or volunteer, or reduce waste output.
Design Statement	effectively create an algorithm that can work for any type/form of image
Constraints	- automated process that works for all images - need a logo or superimposed images, masking/shading - need a parameter the client can control, such as abstract art and geometric shapes - tall image, wide image, square, partly transparent
Deliverables	- functioning algorithm (Python program) - manipulated images - documentation w/ images - slide show showing final images/product

Next, we stated what specific manipulations we wanted to make for our Tier 1, Tier 2, and Tier 3 level product:

Constraint	Met by functions that/for:		
	Tier 1	Tier 2	Tier 3
Automated process that works for all images	• add border to all images • paste logo on images • resize images • run through all images in a folder	Tier 1, plus: • function that pastes text on images	Tier 2, plus: • addition of shading • color manipulation /grayscaling
consistent logo or superimposed images using masking and/or shading	• pasting logo as an image on all other images • pasting an image on a sea creature on the image to be modified • pasting of an image highlighting environmental destruction over one highlighting environmental beauty • pasting a "donate" sign on all images	Tier 1, plus: • blending two images together to produce a "before-after" effect of pollution • shading a pasted image to add a feeling of futuristic demise • composition of multiple images: stacking images vertically on one another, pasting a piece of trash on an already blended image	Tier 2, plus: • highlighting a specific part of an image (increasing opacity relative to rest of image) to draw attention

	Tier 1	Tier 2	Tier 3
parameter a client can manipulate- such as abstract art or geometric shapes	• border color and opacity • curvature of the border • thickness of the border	Tier 1, plus: • opacity of two images as they are superimposed • text that is typed on the image	Tier 2, plus • what pixel size and shape the product image is • creation of a specific geometric pattern on the border
needs to work on images of different shapes and sizes	• selecting at least one wide, tall, and square image • resizing images automatically	Tier 1, plus: • having images of widely different pixel sizes • manipulating a partly transparent image	Tier 2, plus: • dealing with images of different shapes (curved, round, or with geometric cutouts)

Finally, we sketched what we wanted the final (Tier 3) images to look like:



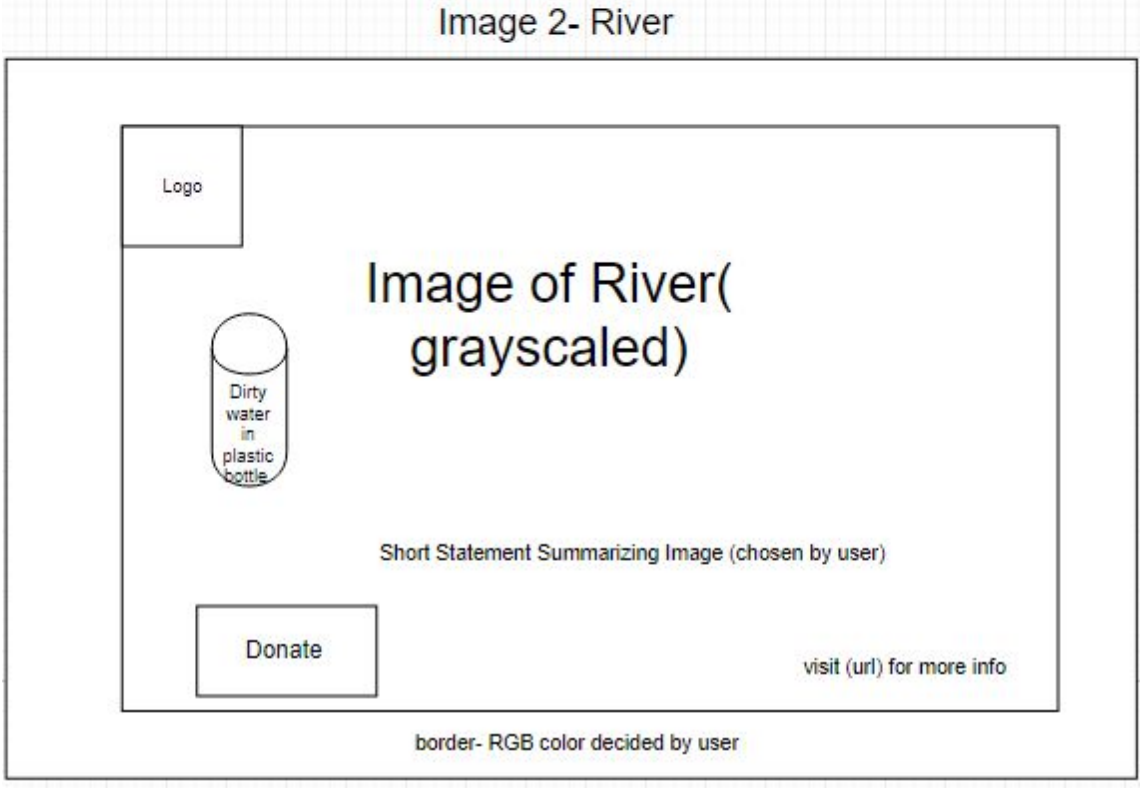


Image 3- Underwater view of boat

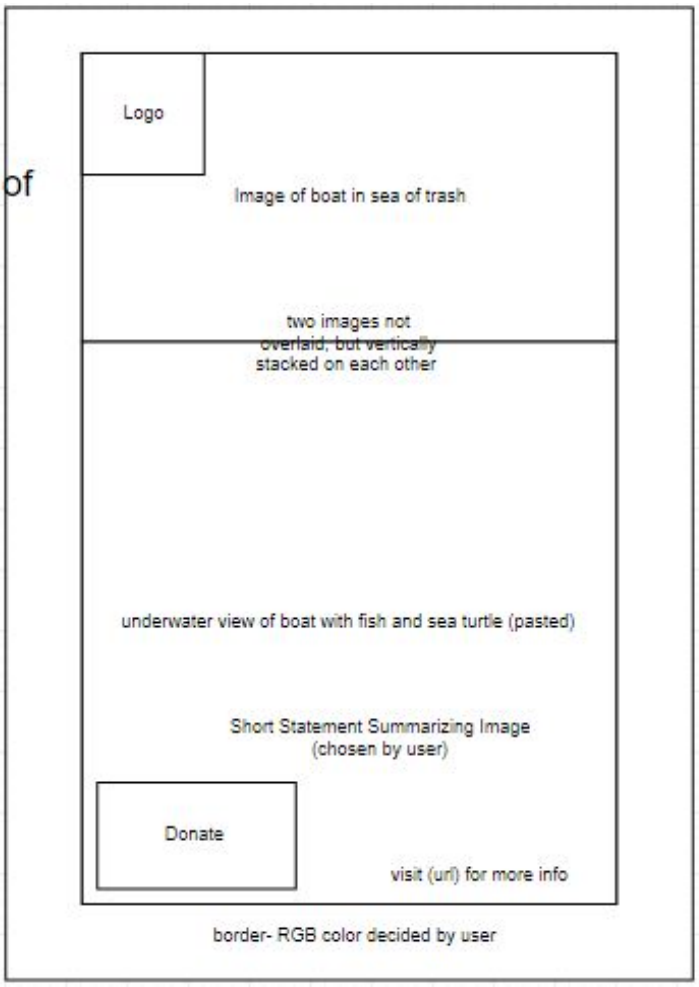
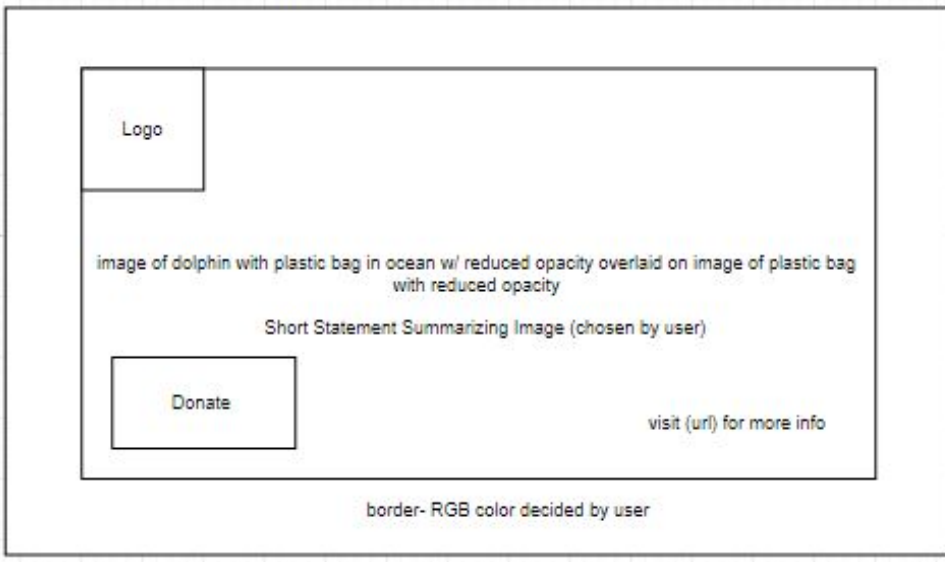


Image 4- Blend of plastic bag in store and dolphin and plastic bag in ocean

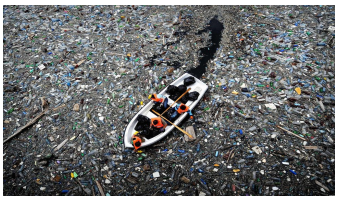


Raw and Final Images

Since the user chooses what text is written on the images and what color their border will be from many different possibilities, we only showed some of the possible combinations below. The user can directly control: image size, shape of border, color and opacity of border, and text on image.

Original Images:



Border Colors: User choice



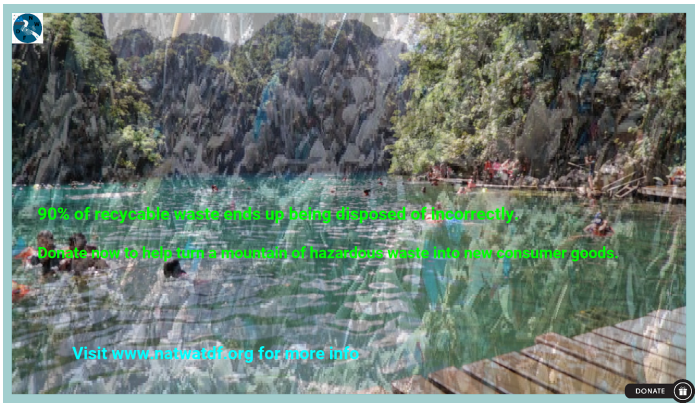

Border Shape: User Choice

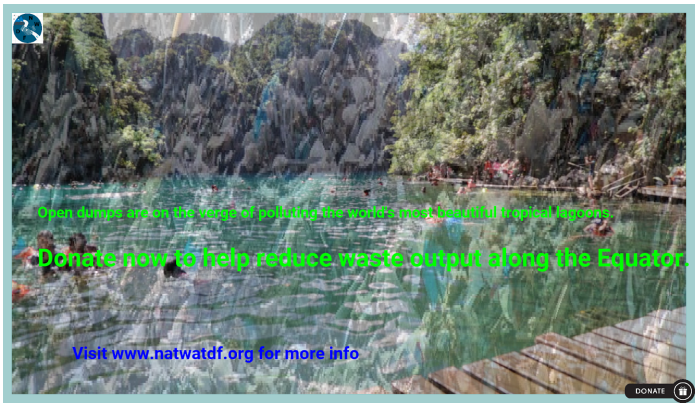



Border Stacking: User Choice




Image Text: User Choice





Gallery Walk Table

Project Cloud 9 Workspace Project Client: Designers: Google Slides Image Artist Presentation	https://ide.c9.io/reachsragvi/csp5-vadali-s-python Natural Waters Defense Foundation Loic Scomparin & Sragvi Vadali NWDF Image Presentation

Instructions: Open a new terminal. Go into a python workspace by typing "ipython". Then go to the folder Final_images by doing "cd 1.4.7/Project_Images/Final_images". Final run the code using **"%run ../../final.py" Ensure when running the code that there are only 4 items in the folder**

Likes/Pros	Missing/Cons
1. I like the way how you pasted 2 images on a canvas and made them blurry, so that you can both the images really clearly 2. The messages are actually meaningful x2 3. The pasting of the picture of the plastic bags on the pic of the dolphins is clever and creative 4. Could actually be used for pictures for a campaign against pollution 5. nice job showing the changes you made to the images	1.Slides are kind of messy 2. The terminal does not prompt me with instructions 3. the slides goes to fast making people couldn't see the words on the slide 4. The slides are hard to read 5. The messages in the pictures are also kind of hard to read sometimes 6. Some docstrings could be more descriptive 7. They left commented code in 8. The colors of the text could have been toned down to match the images (x2) 9. Nested functions are not the best practice; algorithms and code are redundant 10. On the slides, the client description could be neater by cropping out the document footer 11. The logo on the images should be transparent for visual appeal; try masking 12. Try to stay consistent with font on your documentation (don't switch between two) 13. Try fixing the extra spacing in the documentation and remember to delete the instructions 14. Add a qr code of the user's choice maybe? 15. the background of your slides make it hard to see the frames you added

Conclusion

Some shortfalls of the project (FYI Mr. Brown):

The project instructions mentioned that the client wanted an automated process to apply to all the images, which we did achieve with regards to pasting the logo and the donate sign. However, we wanted to augment the impact of our images by retaining a certain level of creativity in each one, so some of the code is specific to a particular image, such as blend() for the dolphin image and the grayscale code for the river image. The code for these modifications is organized into separate algorithms, however, note that *we found it impossible to apply all the code to all the images at once*. There, we *pre-ran the code so that the user only has to apply the finishing touches (text, border, logo) to the image*. The user also chooses what text to place on what image, so the same text will also not be applied to all images at once.

Also, there are limitations of the way we solved some of the problems we encountered. For example, we were unable to figure out how to type text and have it go an a new line when it was too long, so we made a math function to make the font smaller as the text got longer to keep it on a single line. This obviously isn't a perfect solution and if the text typed is extremely long, it will be too small to read and/or go off the image.Yet another example is when we resized images for blending- a tall image can be blended with a wide image, but we didn't crop the images to make the result look better.

Finally, we are having difficulties regarding the client-controlled parameters. The client can change the type of border (curved or rectangular), and change the border color and opacity, as well as resize the image. However, the pasting of the logo and donate signs is hard on resized images and they don't appear in the right locations all the time, and we couldn't access the text file to let the user type their own text on the image (as of 3/28), which we hope to resolve.

Loic's conclusion:

The team dynamic was efficient but could use some improvement. Overall, the delegation of tasks was fairly efficient with Sragvi working on the code for blending and gray scaling, while I worked on the code to add text to the images., as well as the project documentation. We also both worked to run code and give feedback to the other who was working on the code. It seemed like communication was sometimes lacking, however, as Sragvi and I didn't clearly specialize what needed to be worked on as a priority. I was more focused on sticking to the design plan whereas Sragvi created and tried new code he hadn't talked about previously. Ultimately both of those actions led to positive ideas for the project but the whole process would have been more streamlined had we better communicated our intentions to each other.

The design process was also fairly efficient, but there are key areas for improvement. We spent too much time early on the project identifying our client and images to use for the project, leaving us with less time to tackle the more advanced modifications later on. We should have also made the code more pertinent to our goals earlier on. For example, we didn't change the border function to allow for raw input at first, and that caused many errors when we tried to adjust the code later on. Our brainstorming plans were also lacking in detail in terms of the specific modifications we wanted to do for each image. However, one thing we did well was test code relatively quickly on a sample image before revising it and applying it to all images. We also did a pretty good job of recycling code from 1.4.4 and 1.4.5 to save us some time and not create the code to access and modify images from scratch.

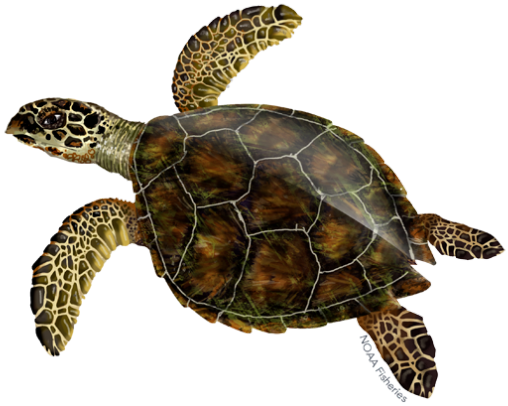
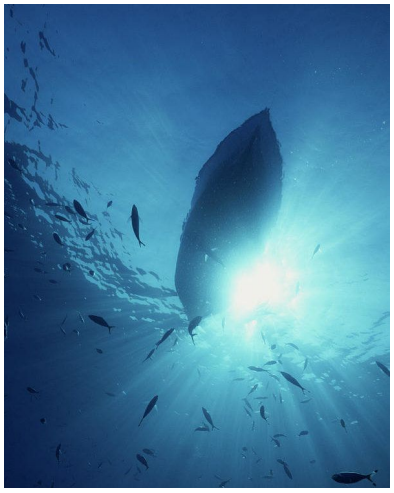
Sragvi's conclusion:

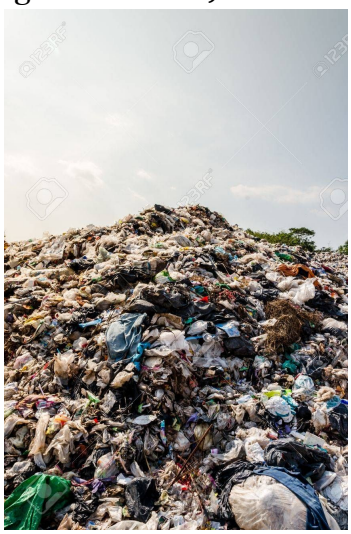
The team dynamic was fairly good. Loic and I were able to communicate with each other efficiently in order to finish the task at hand. This allowed us to be able to finish the work and learn from each other. With Loic's prior knowledge in engineering, I was able to learn new key concepts like a design process, an efficient way to brainstorm our ideas. As well, I learned many concepts in Environmental Sciences. We were also able to develop on each other's strengths. Since Loic is a senior, he has more research skills than me, causing it it be easier debug and create new ideas. Ultimately both of those actions led to positive ideas for the project but the whole process would have been more streamlined had we better communicated our intentions to each other.

The design process was fairly simply. We started the project by creating a brainstorming chart and the constraints of the project. This allowed us to be able to imagine the code in our head. After that, we created a template on how we wanted to do the images as well as what the images will look like. We were happy with our templates and started working right away. With Loic's prior knowledge, we were able to create a strong basis for our images and what we need to work on. However, we made the code to complex that we didn't understand a way to perfectly streamline the code. This made it a little more challenging. However, we were able to use the design process and create a good image manipulator.

Daily Project Log

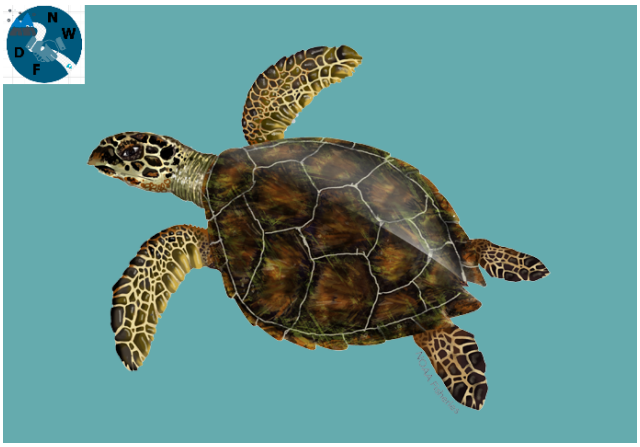
Date	Description	Images
3/4/19	We chose project Client 1 - a Cause- and brainstormed possible causes/groups that we could represent: - education - politics - humanitarianism - philanthropy - economy - environment We chose environment as our topic, and upon further discussion, chose *pollution in natural bodies of water* as our specific topic.	None yet.
3/5/19	Going off of the topic we defined yesterday, we decided that the main goals our client organization wanted to achieve was promote donations to their site to help fund education and cleanup efforts, and also urge people to produce less waste, recycle, etc. We then defined our client, the fictitious national organization Natural Waters Defense Foundation. Finally, we summarized all that we had done so far into a design brief: Chosen idea: Environment - specifically water pollution Client: Natural Waters Defense Foundation Client Objective: advocate for proper disposal of trash/no dumping in oceans and environmental cleanup of trash that is already there. Problem Statement (issues we need to address): - illegal and legal dumping of trash and chemicals in waterways and surrounding land - create and manipulate images to effectively show the issue and prompt people to donate or volunteer, or reduce waste output. Design Statement: - effectively create an algorithm that can work for any type/form of image Constraints: • automated process that works for all images • need a logo or superimposed images,	None yet.

	masking/shading • need a parameter the client can control • abstract art and geometric shapes • tall image, wide image, square, partly transparent Deliverables - functioning algorithm (Python program) - manipulated images - documentation w/ images - slide show showing final images/product	
3/6/19	Now that we had our client and purpose down, we looked into different ideas of what we could portray to help gain the public's attention to the cause: Ideas for all four types of image (tall, wide, square, partly transparent) - general idea: paste a sea creature in image of environmental ocean pollution - tall image: both above and underwater views so people can see things from an animal's perspective- boat above water dumping and trash floating underwater - wide image: lagoon with people bathing and open landfill in the background - partly transparent image/square image: before and after pictures of disposable products/waste (overlaid transparent images) - other wide image idea: upstream half of river is clean and downstream half is full of trash, would need to combine two different image or overlay a transparent image on part of the original image We began to browse the internet for images that would fit these ideas, as well as images we could use as masks. Concurrently, we began brainstorming features required for all images that we needed to develop: • a border- possibly a rectangular one, to emphasize the seriousness of the situation. The border could be from another image, such as that of water, cut out using a mask. • a consistent "donate, volunteer, etc" statement • a fake url for the website of the organization	Unmodified images for the project found so far: turtle that could be pasted on other images:  underwater view of boat (tall image):  lake (wide image):

	• a logo (which we still needed to design) • a geometric shape in the border (maybe some smooth curves representing waves, although that may be hard to do • a short, concise, provocative factual statement for each image, placed before the "donate" and url features. Eg. *The Great Pacific Garbage Patch contains around 35 million tons of trash*. Develops the ethos/logos aspect of the image. • possibly a sea creature- such as a fish or turtle- that could be pasted on each image and "say" the statement. This would help develop the pathos aspect of the image.	
3/7/19	Since creating the border (with statements and logo) for each image was our utmost priority, Sragvi began manipulating the 1.4.5 code to see if he could develop a rectangular border. Loic began creating the company logo in draw.io. Both of us also found additional base and masking images we could use for the project.	New Images Found: trash dump (could be pasted in background of lake image seen above):  Polluted river (wide image):  Ex. of modified image with new border, from

		1.4.5 (now unsure between using something like this or rectangular border). 
3/8/19	Loic finished drawing the company logo, and saved it as an image and uploaded to Cloud 9. Sragvi found additional images off of the Internet that could be used for the partly transparent image suggestion. Both of us worked on a .txt file in C9 to document the sources for all our images, for future reference. At this point, it seemed like we had all the components of our project ready and we needed to start figuring out how to code it all together: • using online documentation, we examined how the blend function (combine two images by altering the alpha value) was used. link: https://pythontic.com/image-processing/pillow/blend • Sragvi also figured out how to create a rectangular border using the 1.4.5 code. This will be useful later on as we refine our border.	New Images found: bag in store and dolphin with bag in ocean: We could overlay these images to demonstrate where plastic bags often end up after they are bought, to encourage users not to use them in the first place:  

		Finished company logo: 
3/10/19	We researched new modifications we could do to add a greater impact to the message of our images. Specifically, we looked at how to grayscale images (making them white and black by iterating through the color values of every pixel), and add text features using the ImageDraw class. Since we decided that we would type text directly onto the images, rather than the border, this class will allow us to experiment with a variety of text locations on the image and pre-downloaded fonts. Third-party documentation on how to grayscale images: https://stackoverflow.com/questions/12201577/how-can-i-convert-an-rgb-image-into-grayscale-in-python	No new images.
3/11/19	We both worked on figuring out how to add a symmetrical border around all the images. We finally managed to do so by creating an image object larger than each of the images by 25 px on each side and pasting the original images onto the border. We had multiple failed attempts because we did not realize that we had to use the paste function to paste the images at a specific location on the border image object, and then we tweaked the numbers to achieve the thickness border that we liked.	Example of an image with initial border manipulation:

	Additionally, Sragvi worked to resize the logo file and paste it on all the images in the Project_Images folder, at a specific pixel value. Since this first main algorithm met the minimum viable product requirements, we saved a backup version of the code. *One thing we need to work on later is getting the border to be proportional to the length and width of the image, rather than using a constant pixel value, so that it does not appear thin on large pixel images and too thick on the smaller ones. Similarly, we need to paste the logo on a location proportional to image height and width.*	 The border is not the same thickness on all sides of the image. After several iterations, we arrived at a result with relatively even borders on all images:  That said, the border seems very thin on a large pixel size image, as seen above.
3/12/19	Today's main focus was refining the functions used to apply the border changes to all the images, as well as organize the file system. We removed some folders and images that we had created as tests, but that did not satisfy the border criteria we were looking for. Loic reviewed the 1.4..4 and 1.4.5 code to figure out glitches and errors in the functions and help Sragvi clean up the code. One significant task was figuring out which of the masking-related code from 1.4.5 was still viable to use in our project. By the end of today, we were able to make a successful template function that could open all the original images in the Project_Images folder and alter all of them, using the 1.4.5 code Some links that proved useful to us in deciphering the various errors we had encountered after the past couple of days:	An example of an image with the logo pasted at the upper left corner (it overlaps both border and image): 

	http://cs.carleton.edu/cs_comps/1213/pylearn/final_results/encyclopedia/attributeError.html https://teamtreehouse.com/community/in-python-whats-the-difference-between-a-type-error-and-a-value-error We were confused between type errors (like using int instead of string) versus value errors (incorrect argument of the correct type, more specific than type errors).	 Rough version of code used to create the border for all the images.
3/13/19	We didn't meet in class today, but both Loic and Sragvi researched how to combine two images using the blend() method. We discovered that blend() interpolates the colors of the corresponding pixels in each image to produce a new image that appears to be a combination of the two faded out original images. Which image is more visible can be controlled by using an argument value ranging from 0-1.0. Blend() documentation found online for future reference: https://pythontic.com/image-processing/pillow/blend https://stackoverflow.com/questions/29106702/blend-overlapping-images-in-python	No new images.
3/14/19	Today we both wrote the blend function to overlay two images. We wanted to overlay the image of the plastic bag in the store with that of the dolphin and the plastic bag in the ocean as described under the Images column about a week ago. We also encountered some attribute errors while trying to write the blend function to suit our needs. Those were caused by entering nonexistent arguments for blend() in an attempt to specify in what matter and what location we wanted the images to be interpolated.	Initial version of the dolphin/plastic bag montage:  The image of the dolphin was blended with

	One minor issue we need to fix later is that we coded the blend function to take one image and interpolate it will all the images in a folder, whereas only one other image needs to be blended with it. We know how to remove this, but we wanted to see whether the function worked for all images, even those of vastly different shapes and sizes. *As seen to the right, it did "work" for all the images (the image of the dolphin blended with all the images), but width and height are definitely an issue. We will probably need to resize a lot of images later on…*	all images, including itself (didn't look any different since we didn't resize it)  Non-ideal blend example.

3/15/19	Today we revised the blend() method by changing the opacity of the blend to achieve the most visible mix of both images. We also reprogrammed the logo to be on the upper left corner just inside of the border of all images.
	
	We also began to further think about how we could blend different images to have the strongest effect for our cause. For example, we want to paste an image of a boat dumping trash on top (not over) the image of the underwater boat view and fish. These are things we will do at the end of the project timeline if we have time, but for now we will focus on the minimum viable product constraints.
		
		Example of image with logo in the corner.
		
		`def algorithm_4():`
		`  def watermark(original_image, alpha, image):`
		`  """`
		
		`  """`
		`  width, height = original_image.size #gets dimensions of target image`
		
		`  image = image.resize((width, height))`
		
		`  result = PIL.Image.blend(original_image, image, alpha)`
		
		`  final = PIL.Image.new("RGBA", (width + 50, height + 50), (101,171,174,150))`
		
		`  final.paste(result,(25,25),mask=result)`
		
		`  return final`
		Final code to blend two images together.

3/16/19	Loic- caught up on documentation.	No new images.
3/18/19	Sragvi worked on algorithm_2(), which had already been created to create the border of the function. He added code to paste an image of a turtle on top of all the other images. This will prove to be useful later on, as having a superimposed turtle on all the images provides a consistent incentive for people to donate and volunteer to keep waterways clean.	 Lake image with resized turtle image pasted at a specific location.

3/19/19	Loic further researched how to add text to images. There is a built in class called ImageDraw in PIL that allows the user to predefine or raw input text directly on an image at a specific coordinate location, and choosing the size, font, and coordinate location of first letter of the text. Before building the function, Loic imported ImageDraw and ImageFont from the PIL library. He also uploaded the Roboto-Bold.tff file, which contains the code need to convert the text to the Roboto font in bold. Therefore, he found all the tools necessary to actually begin building the code tomorrow.
	
	Sragvi worked on algorithm_3(), specifically the code to grayscale images. This will most likely be used for the image of the river, since all the other images already have major modifications (blended or pasted images). The grayscale image will appear starker to viewers and help them grasp the severity of litter and other aquatic pollution.
	
	Sragvi encountered type and value errors. They came from incorrect assignment of variables pertaining to the application of a mask image and not iterating through the rows and columns of pixels in the image in the correct order.
		No new images.
		`for n in range(len(image_list)):`
		`  print(len(image_list))`
		`  filename, filetype = os.path.splitext(file_list[n])`
		`  # Round the corners with radius = 30% of short side`
		`  image = image_list[n].convert('RGBA') #converts to same format as red`
		
		`  new_image = watermark(image,.4)`
		`  new_image.paste(logo_small,(25,25),mask = logo_small)`
		
		`  img = PIL.Image.new('RGBA', (columns, rows))`
		`  grayscale_img = new_image.resize((rows, columns))`
		`  img = img.paste(grayscale_img, (0,0), mask = grayscale_img)`
		`  image_2 = np.array(img)`
		
		Sragvi's progress on the grayscale code.
		
		Source for the Roberto font code: https://github.com/google/roboto

3/21/19	Loic continued to work on the algorithm_5(), which will add text to all the images. He completed the code so that text could be added anywhere on a single image.
	Here are some issues that need to be addressed:
	• adding text to all images at once, but also adding different text to different images
	• making sure text doesn't just get cut off at the edge of the image and instead forms a new row
	• placing the text at a specific point on the image relative to the size of the image rather than on the same coordinate point for all images
	
	Sragvi continued to work on algorithm_3(), the grayscale. After figuring out how to coordinate using both the Numpy and PIL libraries, he continued to work on assigning black and white colors to pixels by using the average of the sum of RGB values for a certain group of pixels.
		for row in range(len(image[0])):
		for column in range(len(image)):
		average_rgb = image[row][column][0] + image[row][column][1] + image[row][column][2]
		average_rgb = average_rgb //3
		image[rows][column] = (average_rgb, average_rgb, average_rgb, 255)
		Progress on grayscale code.
		
		No new images.

3/22/19	We discussed what we needed to finish during the weekend to have a quality product ready by Monday: Loic Sragvi Both Things to do: CODE • river image: grayscale and paste bottle • pasting images (trash behind lake, donate sign on all images) • text for all images (raw input?) ○ ideas for raw input ■ border RGBA values ■ folder name ■ text ■ what they want to modify • combine image of boat dumping trash (still need to find) above image of underwater view of boat with pasted turtle ○ if time doesn't allow for this we'll just paste some trash around the turtle or do a blend like for the dolphin • clean up all the image folders that will not be used ONCE the documentation is done because need to use some of those images for the documentation • adding the code to iterate through and modify all the images to the new algorithms (alg 5, etc) • comments and docstrings for all functions • new named backup versions of code OTHER • finish documentation • slide show Loic worked on the alogorithm_5() (adding text to images) and tried to create a way to place the text at a location corresponding to a fraction of the length and width of the image, to be more flexible with regards to different sized images. Sragvi helped him by defining global image size variables, but without success. Sragvi worked on his grayscale model and finished a preliminary version (see first grayscale image at right)	 Grayscale image of lake  Preliminary result of algorithm_5().

3/23/19	Sragvi set aside his original grayscaled image code and successfully develop a newer, much more simple version using convert.("LA"), converting the image from the RGB system to black and white.
	
	In order to address the need for raw user input, Sragvi also created a global function allowing the user to choose their image border color using the RGB system.
	
	Loic worked on algorithm_5(), which focuses on placing text on the image. He found a way to place the first letter of the text at a location corresponding to a fraction of the length and width of the image by defining image width and height as equal to the size of the original image. He also found a way to avoid text running off the image by using the image_width variable to make the font of the text smaller if the typed message was over 60 characters This keeps the text on a single line- we couldn't figure out how to place it on multiple lines- without running off the edge of the picture. It is not perfect and really really long statements will eventually run off or lead to a font size too small to be visible.
		Sragvi's final grayscaled image:
		 
		filename, filetype = os.path.splitext(file_list[image])
		# Round the corners with radius = 30% of short side
		image = image_list[image].convert('RGBA') #converts to same format as red
		
		new_image = image
		
		new_image.resize((1000,1699))
		new_image = PIL.Image.open('1.4.7_river.png').convert('LA')
		final = new_image
		final.paste(water_large, (int(float(columns) * 0.75),rows - 700), mask = water_large)
		final.save("final3.png", format = "png") #saves img
		
		Example of text of different lengths:
		 
		 

3/24/19	Loic finished working on the text algorithm. This consisted of tweaking the math function controlling the font size based on the number of characters in the text, adding another lines of text, and making two of the three lines of text raw input. He then tested the algorithm on different text lengths and image sizes. Sragvi cleaned out type and value errors coming from improper arguments in the paste function for the logo and the turtle image. He also began to run the code, one part at a time, to produce the final images, without text (see right). This is necessary because not all images have the same modifications, such as grayscaling, but all have some features, like the logo. Both Loic and Sragvi worked on the code for the above and underwater view of the boat (3rd image from the top at right). The image was produced by resizing both images and pasting them on a white background image of identical width. Loic then ran all the images Sragvi produced through algorithm 5, producing two copies of each images, each with different text. He tried to get the code to let the user change the color and opacity of the border to work, but without success. Loic made the presentation and started working on the cover page and gallery walk page of this document.	final images with text:  

3/26/19	Sragvi attempted to run the code to produce a new set of images to be used in algorithm_5() (since we still need user input for color) but encountered PIL and blend() attribute errors in algorithm_1() and algorithm_3(). Loic examined the code later in the evening and tried to troubleshoot the errors, but also without success. Loic continued working on the documentation, focusing on the references and brainstorming pages.	No new images.
3/27/19	We both worked on the code to let the user add text and borders to the image, Sragvi writing the code in a separate file and Loic testing the code and providing feedback. We still ran into errors and were unable to change the border on the images by user input. Loic also worked to update the project log and the rest of the documentation, including designing the cover page and array of raw and final images.	None new.
3/28/19	Sragvi continued to work on the code to allow user input for border design and text input. The code was not able to access the text file needed to write the text on the images, so we focused on the border instead. We got several user-induced examples of different borders (colors and shapes), as well as different sized images, but were plagued by the logo and donate symbol appearing on the wrong place on the images or not at all. As of now we are still working to resolve this issue.	Examples of images with reformatted borders:  curved border

		 user-chosen border color

Credits for Raw Images

Image name	Reference
1.4.7_dolphin-bag.png	Retrieved from https://i.dailymail.co.uk/i/pix/2014/09/25/article-0-215CF7010000000578-345_634x286.jpg
1.4.7_turtle	Retrieved from https://cdn2.webdamdb.com/1280_EdNqFCk8lR98.png?1539707027
1.4.7_underwaterboat.png	Retrieved from https://photos.com/featured/fish-swimming-near-boat-underwater-view-zena-holloway.html?product=poster
1.4.7_baginstore.png	Retrieved from https://www.wdbj7.com/content/news/Kroger-starts-reducing-plastic-bag-use-491580581.html
1.4.7_river.png	Retrieved from https://feature.undp.org/plastic-tidal-wave/
1.4.7_lake.png	Retrieved from https://d2v9y0dukr6mq2.cloudfront.net/video/thumbnail/N0ntERZOxijfnsnsl/people-swimming-and-diving-in-a-volcano-lake-00-11_ssu3dxevx_thumbnail-full01.png
1.4.7_trash_pile.png	Retrieved from https://img.huffingtonpost.com/asset/57a4d3e72a00002d004f89e2.jpeg?ops=scalefit_720_noupscale
1.4.7_Donate.png	Retrieved from https://www.google.com/url?sa=i&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjR-LC7-I7hAhXTv54KHdF1CysQjRx6BAgBEAU&url=https%3A%2F%2Fwww.recoverycoco.com%2Fdonate-button&psig=AOvVaw1PKjHdwLsw58CbbbxZruij&ust=1553110179844444
1.4.7_dirt_water.png	Retrieved from https://www.exilemod.com/wiki/items/plastic-bottle-dirty-water/
Roboto-Bold.ttf	Retrieved from https://github.com/google/fonts/blob/master/apache/roboto/Roboto-Bold.ttf