Journal Report 3
9/16/19-9/20/19
Oliver Hayman
Computer Systems Research Lab
Period 4, White

Daily Log

Detail for each day about what you researched, coded, debug, designed, created, etc. Informal style is OK.

Monday September 16

There was an issue with a method which converts a matrix to a form that gives the thickness of an object at certain positions (i. e., the matrix [1, 1, 1, 1, 0, 0, 1, 1] would be converted to [4, 3, 2, 1, 0, 0, 2, 1], since there are four 1s before empty space at the first position, 3 1s at the second, etc.). The method it turns out edited a matrix parameter of the method, when I intended it to return a new matrix while leaving the parameter unedited. In the method, I assumed that for a matrix m, "m[:][:]" would return a deep copy of the matrix, when it in fact does not. I replaced this with a for loop that goes through the array and creates a deep copy of each element, and appends it to another array. This fixed the issue.

Tuesday September 17

Matrices that represented the packing of objects ended up packing them slightly too high, so I went through and edited the packing method. It turned out that I had mixed up several numbers representing rows and columns (for example, I kept getting an out of bounds error because I put mat[0] as the number of rows to be tiled instead of mat, so I had to go through my code and change every occurence of this). I also realised that an initial position I created for packing objects was incorrect (was too high, hence why objects were hovering), so I edited this position to account for this error.

Thursday September 19

In order to create some sort of temporary visual representation of the packing other than just a matrix, I made an ImageCreator Method. Inside this method, there is a map of integers to 3-tuples, which represent different RGB Colors. The method loops through all numbers in an array and converts each distinct number to a color coded pixel, where if a number has no given color, a random color will be given to the number in the map, and if there is an entry in the map, it will be colored the corresponding color. This creates an image of the packed object such that each object has a random color assigned to it.
Additionally, I created a duplication method. What this method does is take a name of an image file and number as parameters, and creates copies of the image file, with the number of copies being specified by the number parameter. This makes it easy to pack multiple of the same object.
I tested out packing and image method for a while to find best possible pixel scale for images. I could make it fairly small and still have the code return packings in a relatively quick amount of time, settled on a small scale of 0.02 (a pixel represents 0.02 feet/inches, depending on size of objects)

Timeline

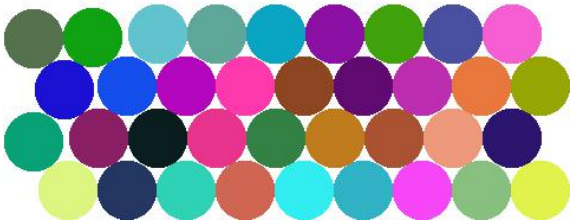
Date	Goal	Met
September 6th	Last week I wanted to be able to generate a back and white pixelated form of an image given an image of an object with a white background	Was able to get accomplished, also was able to convert to matrices
September 13th	I wanted to be able to create several rotations of several images and convert them into arrays using only one method. I also hope to be able to insert an array of integers inside of another array of integers so I can begin the process of using the matrices to represent ways to stack the objects	Was able to insert arrays into other arrays, and created a method that allowed checking if able to do so
September 20th	I hope to be able to specify a column in a matrix representing a packing container and have my code place an object in the optimal position and rotation in this column	I have not made any progress on finding the optimal rotation of an object. However, I have created a working code that packs objects *given* the rotation of each object. I hope to have some sort of successive learning approach to find the best rotation for each object
September 27th	I hope to figure out a method that efficiently finds the optimal rotation of an object by initially testing out several rotations of each object and then progressively gets closer to the optimal rotation through these tests, and begin work on it	
October 3rd	I hope to finish programming the method of finding the optimal rotations for each object found in the previous week	

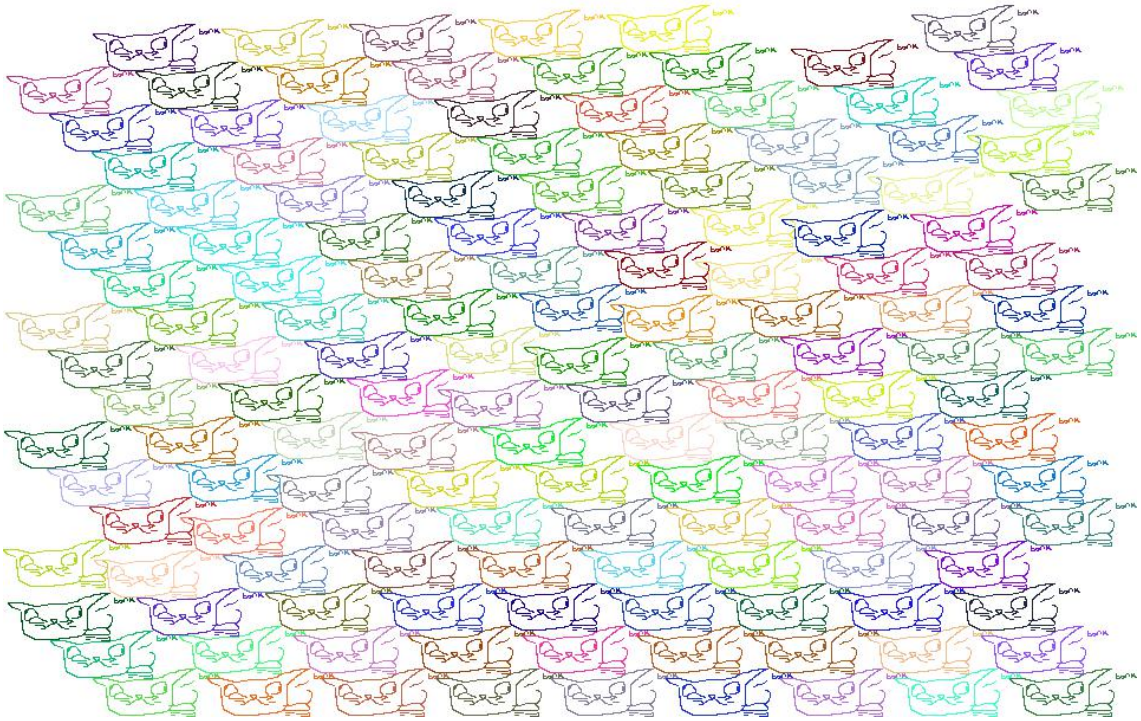
Reflection

This week ended up being less productive than usual, as I had several errors in my code I needed to fix.

One of the largest errors were that objects appeared to ”hover” over the surface of containers in which they were being packed. I found out this was a result of setting the initial location of packed objects too high, so I modified this. I also had to go through my code and fixed several row/column mismatches.

I created a way to represent packings of objects in an image form based on locations of pixels in my array representations of packings. Some of the results are shown below:





I am concerned about finding the least computationally expensive method for the optimal rotations for packing objects. I initially planned to try out different rotations and then try out rotations closer to previous rotations that were optimal to selectively get closer to the optimal packing, but never came up with a specific process of doing this, only a general idea. In my proposal I said I would just create many rotations and try each one when packing each object in a brute force approach, but I realize this is probably too computationally expensive.

In the next week I'm planning on researching different ways to maximize/minimize a function given a set of parameters. One way I think would be to do some sort of modified version of the gradient descent method, as rotations could be considered parameters of a function to be minimized, where the function is the height of the packing of objects. An issue with this is that the function has no clear gradient. I plan to research this method and more in the coming weeks.

Despite setback I had this week, I am happy I am able to at least get some form of a packing algorithm working. Right now my code can theoretically solve problems such as the Ten Penny Puzzle (where ten pennies have to be fit into a square), since the shape of circles are independent of rotation, which aligns with my goal of utilizing my algorithm to solve math problems having

to do with packings.