Journal Report 4
9/23/19-9/27/19
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 23

I began researching gradient descent method and found a way to approximate the gradient of the packing function. I programmed a method for checking the height of a pile of objects, by subtracting the matrix representing a container from the matrix representing a packing of object in the container (removing all of the pixels that are part of the container), and then iteratively checking row by row until the highest row with something in it is found (i. E., highest row that isn't all zeroes in the matrix).

Tuesday September 24

I finished coding the gradient descent method. The code took way too long, so I decided on another method - I would set initially rotations for each object to zero degrees, then check to 120 degrees and 240 degrees, move to smallest height of 3, then from that check current rotation plus or minus 60 and move to smallest, then check current rotation plus or minus 30, move to smallest, etc., stop after a certain point. I did this for each object independently while keeping rotations of other objects constant, so there were only $n\cdot$ (number of steps) iterations of doing this (where n is number of objects being packed), instead of $n\hat{}$(number of steps).

Thursday September 26

Finished coding methon on Thursday. Tried several different sets, but packing method took too long. Shifted focus to doing same method, but once you find optimal rotation for each object, don't pack other objects, which takes significantly longer. This modification took a while as it completely changed my packing method, as initially the rotation of every single object was needed as a parameter, so you essentially had to pack all objects at once and test out all rotations at once.

Timeline

Date	Goal	Met
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	I figured out that the best method for doing this is just by finding the optimal rotation for each object as you pack it, and any specific optimization algorithms would take too long
October 3rd	I hope to finish programming the method of finding the optimal rotations for each object found in the previous week	
October 10th	I hope to be able to speed up my algorithm to the point that I can pack a triple digit number of objects in a class period	

Reflection

This week I spent finding a good method for testing out rotations of the objects. This was met with less success, as I am now at the point in my project where I am modifying a packing algorithm and not just finding an efficient method to code an already existing algorithm (the only modification I've done so far is adding the ability to pack images of objects, and created methods that account for the different scales of each image).
On Monday, I tried specific optimization algorithms, specifically the gradient descent method (using a gradient approximation) to allow for rotation. This took too long. I tried several other methods on Tuesday and Thursday that made it so that I didn't have to modify my packing method, which requires rotations for all objects to be inputted, but all took too long. This means I have to completely redo the method and create a new method of placing objects that allows for different

rotations of objects. This will likely take a lot of time to program, I'm expected at least a week or two. Because none of the methods worked very well, I was unable to get muched accomplished this week. However, I believe I still have plenty of time to modify the algorithm based on my timeline from last year.

I am somewhat worried about the length of time my code is taking, since all of my algorithm modification attempts so far have taken too long, and my biggest obstacle for success is having code that takes too long to run. However, in the future I realize there are two significant modifications to my algorithm I can make to speed things up.

- By changing the height finding method, as you can find the height in O(1) time by just checking it when you pack the final object, instead of O(n) time by checking row by row after all objects have been packed.
- By packing "more pixelated" objects until the optimal rotations are found. An issue with packing larger rectilinear representations of objects is that it increases run time, but if objects are too pixelated the final packing won't represent them well and might not work in actuality. However, if we pixelate the objects more when testing out rotations, and then pixelate them less once a rotation is established as being optimal, this saves time while also keeping the final packing a good representation of the objects

I hope to finish the algorithm and incorporate these changes in the coming weeks. I will likely go through my code and see if there are any other modifications I need to make to speed up my code, as my primary focus right now is saving time. This may involve incorporating other libraries, such as numPy, to speed up certain processes having to do with matrices.