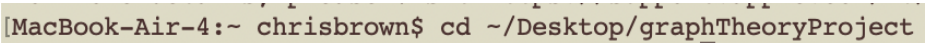
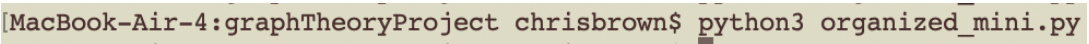


Chris Brown
Ralph Morrison
Math 334
December 12, 2022

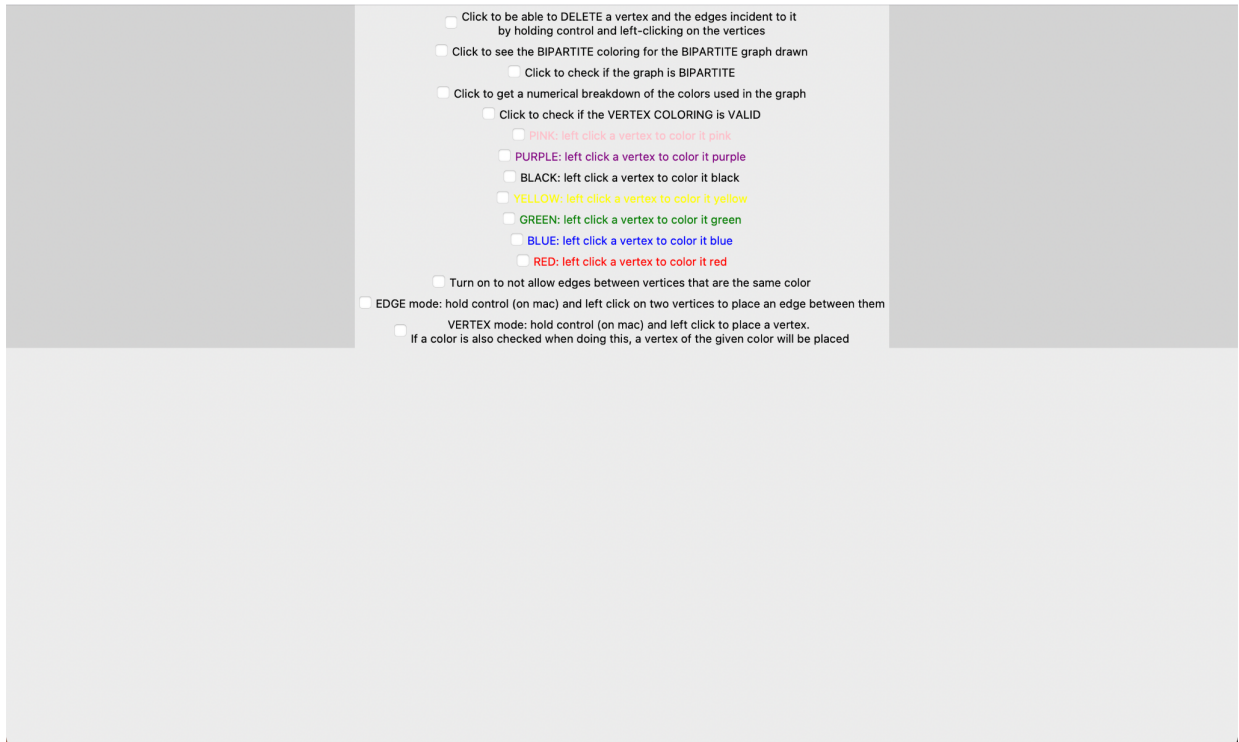
Graph Theory Mini-Project: Graph Creation Interface

Set-up:

1. I installed Python3 and Tkinter using this website, which uses homebrew:
<https://www.pythonguis.com/installation/install-tkinter-mac/>
 - a. homebrew is a package management system that makes installing python3 and Tkinter very easy (just one line commands)
 - b. Here is a link the explains how to install homebrew: <https://brew.sh/>
 - c. Installing Tkinter took about 5 minutes
 - d. Here are the 3 steps required to install the packages needed to run my executable, as explain on the two websites linked above:
 - i. `/bin/bash -c "$(curl -fsSL`
[https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh\)](https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh))"
 - ii. `brew install python3`
 - iii. `brew install tkinter`
2. When everything is installed, download the executable, which I uploaded to glow, and place it in a directory.
3. Open up a new terminal window and navigate to that directory. If you downloaded the executable in a folder called "graphTheoryProject" that is on your desktop, you can navigate to "graphTheoryProject" using the following command:
 - a. `cd ~/Desktop/graphTheoryProject`
 - b. Here is a screenshot:
 - c. 

```
[MacBook-Air-4:~ chrisbrown$ cd ~/Desktop/graphTheoryProject
```
4. To run the executable, type "python3 organized_mini.py"
 - a. Here is a screenshot:
 - b. 

```
[MacBook-Air-4:graphTheoryProject chrisbrown$ python3 organized_mini.py
```
5. The following window should pop up:



Here is a closeup of the options:



Although in my mini-project proposal I said I would be programming in C/C++ and using OpenGL, after spending significant time researching the best way to do this, I determined that it would be a lot better to use python and the tkinter package. The tkinter package is a set of API calls that make using basic GUI tools, input and output, and widgets a lot easier. My executable allows the user to do the following:

- Place vertices of a given color on the screen. The possible colors are pink, purple, black, yellow, green, blue, and red
- Change the color of vertices on the screen
- Create edges between vertices
- Prevent the creation of edges between vertices of the same color.
- Check if the coloring of the graph is valid (no two adjacent vertices have the same color)
- Determine with 1-click if the graph drawn is bipartite
- If the graph is bipartite, draw a 2-coloring with 1 click
- Get the number of vertices of each color that are on the screen at a given moment
- Delete a vertex and all edges incident to it

Reflection:

Overall, I learned a lot completing this project. While I am currently taking a class on computer graphics, it is focused on specific techniques, such as rasterization, cameras and perspective, raytracing, shaders, etc. There is actually a lot of setup required to use OpenGL to draw something as simple as a triangle on the screen. Before attempting this project, I had assumed that there would be a package that would make it easier to create user-oriented graphics applications using OpenGL and C++. While searching for such a package, I often ran into lengthy tutorials. I felt like the setup required to use these packages would distract from the main goal of my project: creating an application that allows the user to create and color a graph,

with other possible features. After spending some time figuring out how to use tkinter, I decided that this was the best package available to me.

I learned many things from the actual programming itself. tkinter is a set of API calls, and getting more familiar with the set of API calls and how they work together took time. The actual organization of the window I created and the use of the `.pack()` function also took some time to figure out. One thing I particularly enjoyed was the modular nature of writing this application. I would add one feature, then another, and then another. Sometimes I would have to go back to a previous feature I created and edit the data structures it used in order to make the creation of a new feature easier. This happened a few times, but one example was changing the representation of the vertices of the graph from a 1D array of circle objects to a dictionary where the keys are specific circle objects, tagged with their xy coordinates, and the values are circle objects representing vertices that are adjacent to their key, which is a circle object itself. This dictionary is essentially equivalent to the adjacency list representation of a graph. New features also constantly interacted with old features in ways that required editing to my old features. This happened after I implemented the call `draw_bipartite()` which uses BFS to 2-color the given graph, provided that it is bipartite. It does so by using the colors red and blue, meaning that it changes the number of vertices of each color. This led to me creating another dictionary called `color_dict` where the keys are strings for each color and the value corresponding to a given key is how many vertices in the graph have that color.

In the algorithms class I am taking this semester I learned that breadth-first search (BFS) can be used to determine whether a graph is bipartite. One of the features of my application is that it allows the user to determine whether or not the graph drawn is bipartite with a single click. The application also has a feature that allows the user to see the 2-coloring of the graph drawn, provided that it is bipartite. When I initially implemented the feature that detects whether or not the graph is bipartite (the call `is_bipartite()`), I only considered the case where a graph consists of just one connected component. Only after testing the calls `draw_bipartite()`, which draws the

bipartite graph, did I notice that I had neglected to run BFS on every connected component on the graph drawn. Modifying my code to implement this change took some work, but it was interesting to implement a version of BFS that runs on every connected component of a graph, as we had always assumed that BFS would be run on just one connected component in my algorithms class. Also, although I had used BFS in various proofs in my algorithm class, I had never actually implemented BFS on a computer, so it was fun and insightful to do so for this mini-project.

Lastly, one thing I found really interesting is that circle objects in tkinter have a tag associated with them, which is a string that can be replaced with an arbitrary string or tuple of strings by the user. I attached a tag that is a 5-tuple to each circle. This tag held the x position, y position, fill color, visited status, and color in the bipartite coloring of each circle. The manipulation of these tags enabled me to add additional features to each circle as necessary, and they were very useful. Their use led to some hard to find bugs though. In one case, I was comparing an integer to one of the members of the tag. This comparison will always be false unless the member of the tag is converted from a string to an integer since tags can only be strings or tuples of strings.

If I could do this project again, one thing I would do is focus first on the organization and features I want my application to have. I think defining these objectives in bullet points on a google doc would have allowed me to streamline the creation of my application. A more straightforward production process could have prevented the often seemingly random bugs that start occurring after the introduction of a new feature due to that new feature interacting in new ways with older features. It would also have prevented the often tedious changes that needed to be made when I changed the data structures I was using due to the implementation of a new feature. Overall though, I had a really fun and rewarding experience completing this project, and I could definitely see additional features added onto it in the future, such as the ability to delete

edges by themselves, rather than having to delete a vertex adjacent to an edge in order to delete that edge.

Also, here is the link to the source code on a public github repository I created:

<https://github.com/cb123450/graphTheoryProject>

Acknowledgements: I used various guides online for help with using `.pack()`, the creation of windows and frames, and other basic features of the tkinter package. All the code was written by me though.