User Controlled Computational Art
Software Design Project 4 Interactive Programming

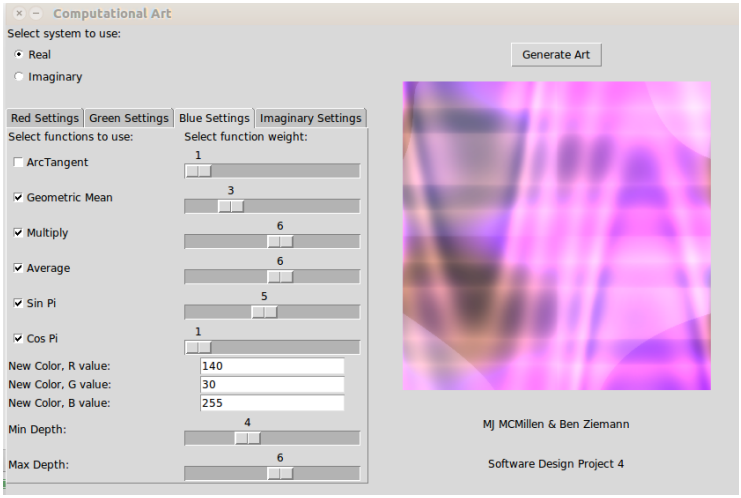
Benjamin Ziemann, MJ McMillen

December 27, 2018

1 Project Overview

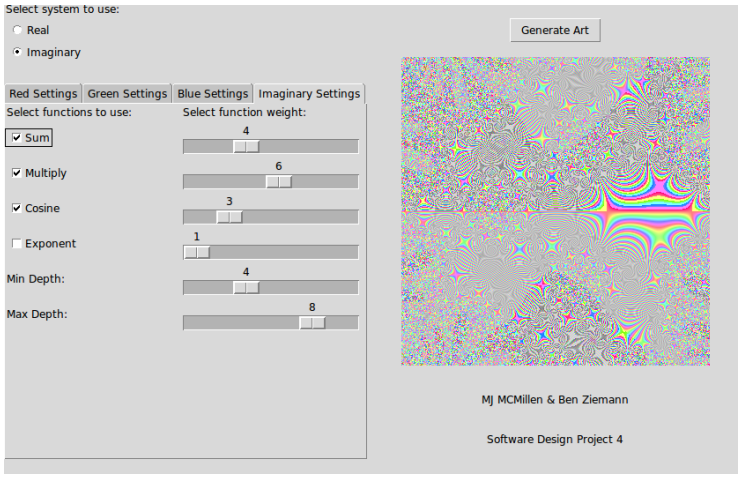
Expanding on our previous project of computational art, we wanted to give the user more control over the color frequency and recursion depth. To do this, we implemented a tkinter based GUI that allows for quick generation of new art. In addition to the GUI, we also implemented increased functionality in the form of complex functions to generate drastically different art.

2 Results



The GUI for Computational Art generating a real number artwork

Above is a screenshot of the result of our two weeks of work. Using Tkinter, we were able to create a GUI that allows for much greater control and access to the settings that go into making the recursive art. The settings are also much more specified, with red, green, and blue each having their own settings as far as which functions use them and their frequency in the project. Each of the colors has its own tab. Additionally these three colors can be overwritten to use other colors instead of the standard RGB.

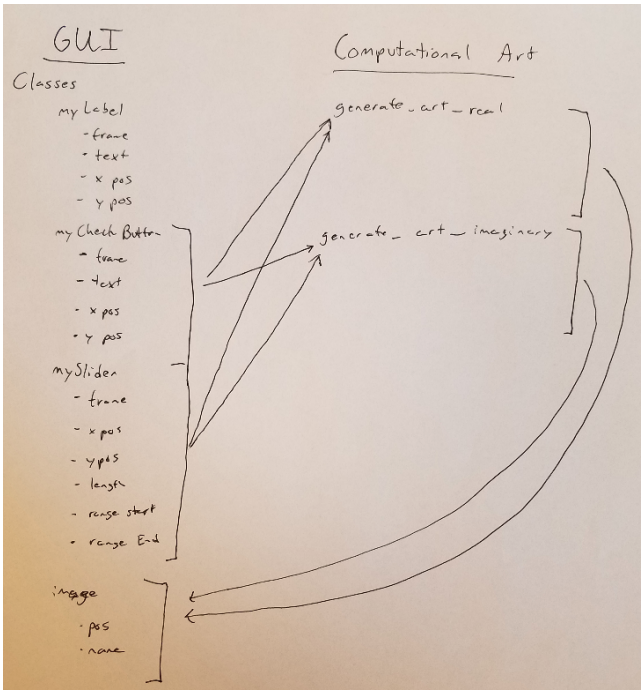


The GUI for Computational Art generating an imaginary number artwork

In addition to the standard the real functions, we also implemented imaginary functions, under the imaginary tab. Like the RGB tabs, these allow for control over individual functions.

Once the settings are chosen, art may be generated by clicking the "generate art" button, which will run our improved computational art and produce a new image.

3 Implementation



UML document for crossover between GUI and Computational Art

The two main components of the system are the GUI and the computational art program that runs behind it.

The GUI was built using the Tkinter library. Although the library had its own classes for buttons, sliders, and other elements used in the GUI, it was much easier and more efficient to create additional class wrappers around each. This allowed for quickly passing in parameters for size, text, and location. Additionally we created additional methods for each class to set the object's position on Tkinter's grid system, which allowed us to quickly and easily change the layout of our project as we added ideas.

The computational art program was adapted from our project 2 code. In addition to that base code, we also implement imaginary functions, using a new generate art function specific to working with imaginary numbers. We used an HSV color system with the imaginary numbers to create interesting art. We also reworked many of the parameters for each of the functions, allowing us to have much greater control and having them able to take in settings from the GUI. For a Real RGB based piece of art, we have three functions dictating color. The user can control which functions are turned on, the frequency they appear, three predominant colors in the art piece, and the minimum and maximum depths for the randomly generated functions. For the imaginary functions, the user can control min depth, max depth, and which functions are present and those functions relative frequencies.

One of the most important decisions we made was the way in which to pass data from the GUI to the computational art program, specifically what type of data the parameters should be. The most difficult among these was the choosing of which function to use with each color and then weighing that against other chosen functions with a slider. We considered passing in multiple lists, a single dictionary, or just a lot of values. Ultimately we went with a dictionary, which allowed us to name the function with the key, and set its weight with the value. We were able to work around the need to see if the function was chosen by simply taking the product whether it was checked or not (1 or 0) and the slider. Thus if the value was a 0, it is assumed the function was not chosen. The dictionary allowed for an efficient and easy to read way to pass all the data we needed to for this part of the project.

4 Reflection

Overall we accomplished a lot, both on the GUI and computational art side of the project. Looking back though we definitely over-scoped the project in some regards, with one of our original long term goals being able to recreate a photo with a grid of recursively generated art. That being said, the unit testing we did for passing parameters and implementing new settings worked well, being able to take it a piece at a time before moving on to the next. This also worked well with our teaming strategy as once we got a part working on just the GUI or computational art program the next step would be to integrate it into the other part.

As far as teaming went we followed through well on our plan, with MJ working more with the computational art program and Ben with the GUI, while still understanding

what the other side was doing. Due to our schedules meeting was occasionally difficult but we made it work and though we fell short of some our of some of our long term goals, we are satisfied with what we did accomplish.