Project Write-up and Reflection

Sam Eppinger and Hannah Kolano

Project Overview

This project will let people who interact with it create art by using the keyboard to color in grids. There is an 8 by 16 grid on the screen that users can navigate using arrow keys, and change the colors of the tiles to preset colors. The user creates their artistic design and then hit the music button! The music is created from the colors which correspond to instruments, and the rows which correspond with an octave. Our program then cycles through the grid column by column and produces a song.

Results

We created an art pad reminiscent of the 19th century impressionist period with an interwoven allusion to Cubism, the early-20th-century avant-garde art movement. This means we created a grid system that could be filled in with various colors. This grid is maneuverable with a cursor. The cursor also is how one selects a tile to choose the color of. This interactive platform leads to a variety of creative possible outcomes for the user to experience.

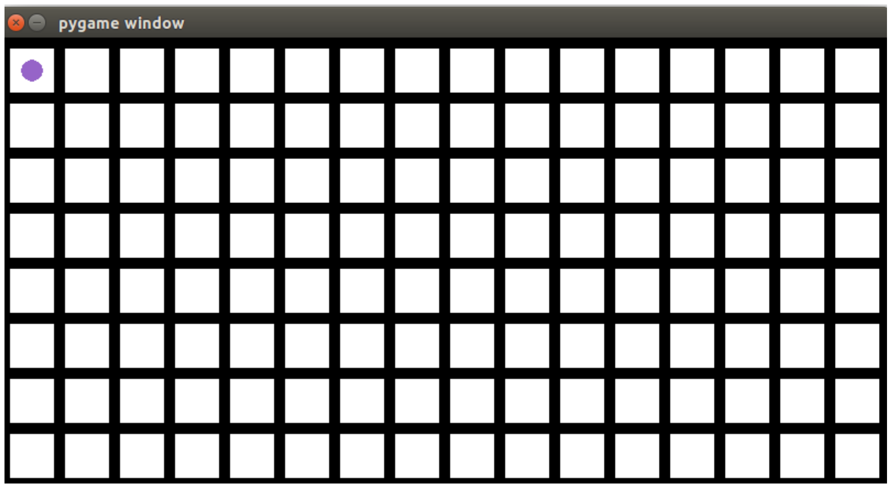

Figure 1: the inital blank grid shown to the user

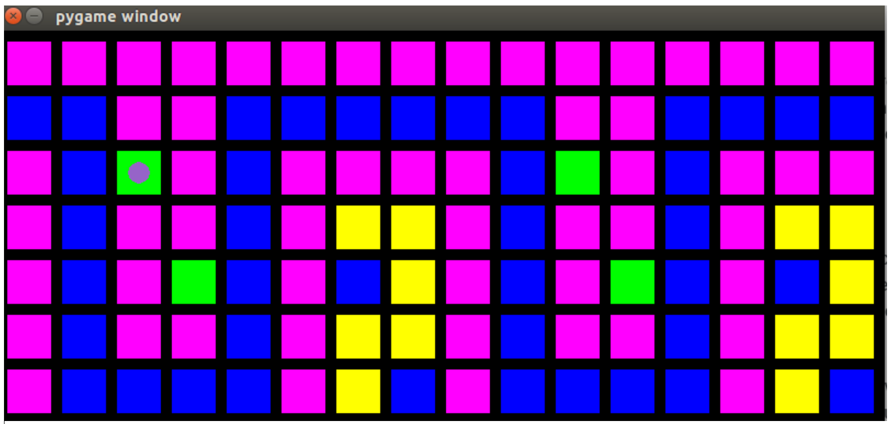

Figure 2: an example of an artistic creation

In addition, our program can also create beautiful, or in most cases an uniquely acquired, tasteful tune. The grid itself plays into the musical part of our program with each row representing a note in an octave. The colors also plays a role, with purple representing cello, yellow representing sax, blue representing trumpet, and green representing guitar. There is a slight pause in playing between each column to create a more melodic song. A problem we came across when creating our song is the audio files we downloaded. Some of the audio files start at different times and some of the notes are tuned to A440 while others are tuned to A441, meaning they do not quite resonate together. Additionally, the sound in each file starts at a different time, causing an ugly overlay of notes. We recommend using the music function with only one instrument.

Implementation

Our program had a number of classes. At the core is the TileArtModel object. It contained the cursor, the grid, and the channels for the music to be played on. The model contains a list of columns; each column contains a list of tiles, which each has its own size, position, and color. The cursor is its own object with similar attributes to the tile, but it can change position; it also keeps track of the index of the tile it is currently on. The controller affects the position of the cursor and the color of the tile it is on.

On the music side of things, there is a parent object Instrument. The four instruments (guitar, cello, trumpet, and sax) inherit from Instrument. Instrument's main attribute is the dictionary that maps note names to the sound files that should be played for those names.

The method read_column in the model object brings everything together. It reads the color of tiles in a column and calls on the correct instrument for that color. Additionally, it reads the position of that tile and determines the correct note for that position on that instrument. It then uses Channels to play multiple sounds at the same time. It moves column to column at a given speed and plays the column's notes.

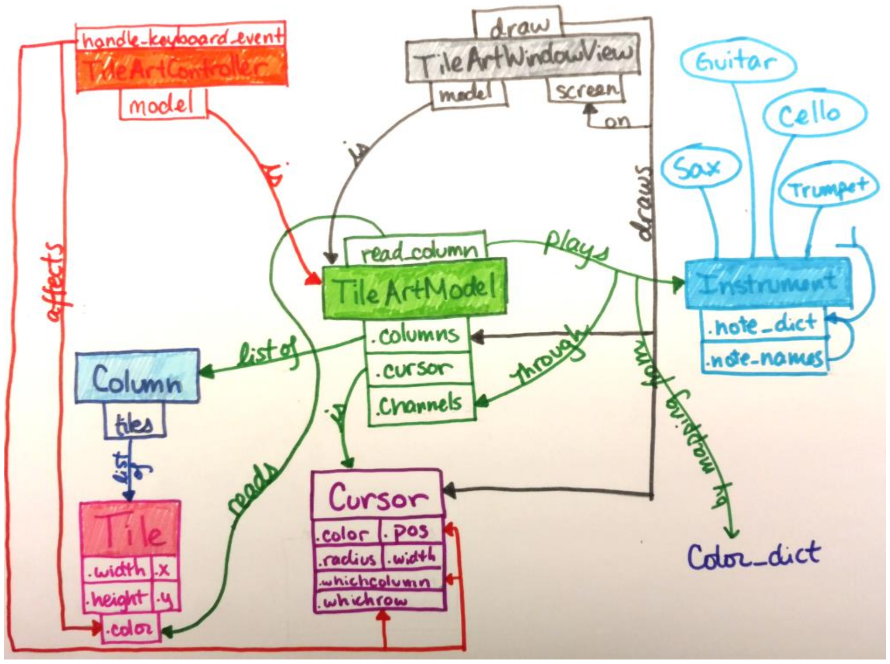

Figure 3: UML diagram of the program

The major design decisions mostly revolved around the user interface. At first, Hannah wanted the spacebar to toggle through colors, but Sam pointed out that it would be easier for users to not have to flip through all the colors to get to the one they wanted. Therefore, we implemented keystrokes instead. Another design involved how the notes would actually be read. At one point we were thinking that each tile would represent a beat, and its RGB value would determine the pitches of the three instruments mapped to red, green, and blue. Since there would be 8 pitches over a span of 255, each pitch would have a range of about 40 points, the lowest note corresponding to values of 0-40, the second lowest to 40-80, and so on. We ultimately decided against that in favor of a system that better visualized pitch: having each row represent a pitch.

Reflection

From a teamwork point of view, we split up work went very well. We were able to work on different parts of the project simultaneously and efficiently while consulting each other. For example, while Sam worked on creating a cursor that could move around the screen, Hannah worked on creating the grid. We then combined the two and worked together on getting the cursor to move solely from tile to tile. After that, we set specific times to meet and work on code together. We seemed to be infinitely more efficient that way. We ran into problems trying to merge our code on Github when we had both made changes. In the future we should figure out how to deal with merge conflicts instead of copying and pasting new code in a separate file.

From a coding standpoint, our process also served us well. We sectioned our code into a MVP and then a couple other “goals”. This allowed us to work from the base of our code up. This structure also encouraged us to have regular meetings to talk about next steps and how we should change our project, as well as discussion about the best way to structure our code. This was an interesting experience because we got to see how other people tackled a coding problem (and the many different ways we could have implemented this project). We will both use this experience as a point of reflection. It will give us the perspective to take a step back and try to think about how another person would think about how to approach the problem.

Overall, we are incredibly proud of ourselves. Neither of us has much coding experience, so to reach our MVP and then go beyond was really fulfilling and surprised us a bit. Even though there are a number of improvements to be done (a bigger grid, more colors, getting the music in tune and playing at the same time), we’ve come a long way, and we’re proud of what we created.