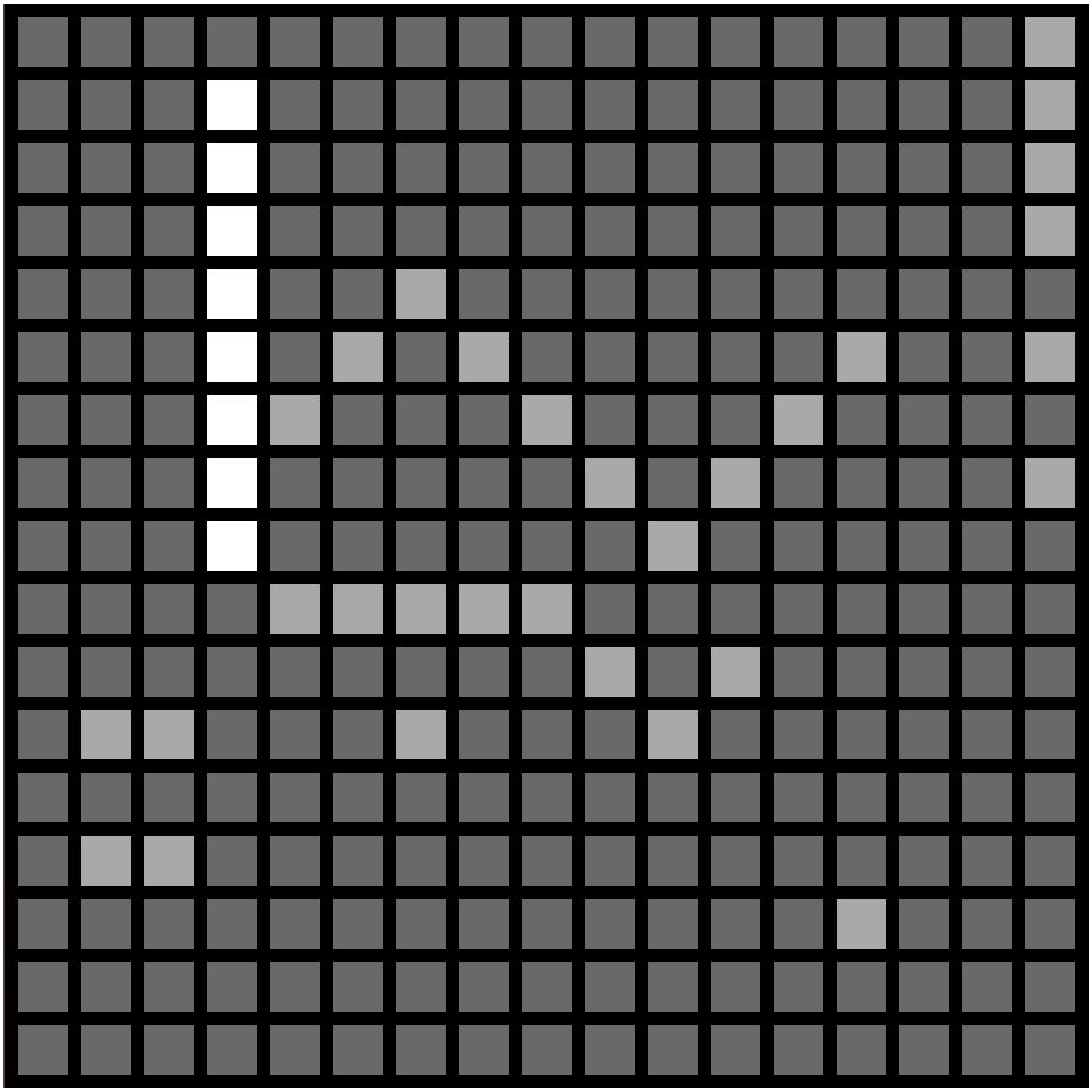


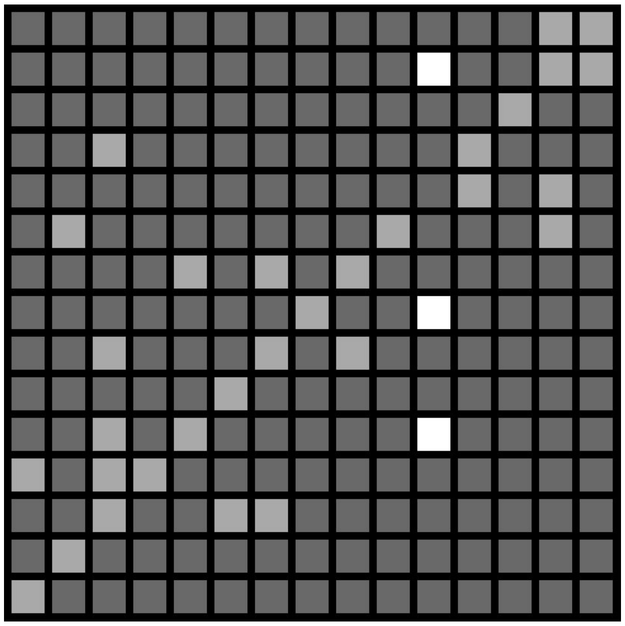
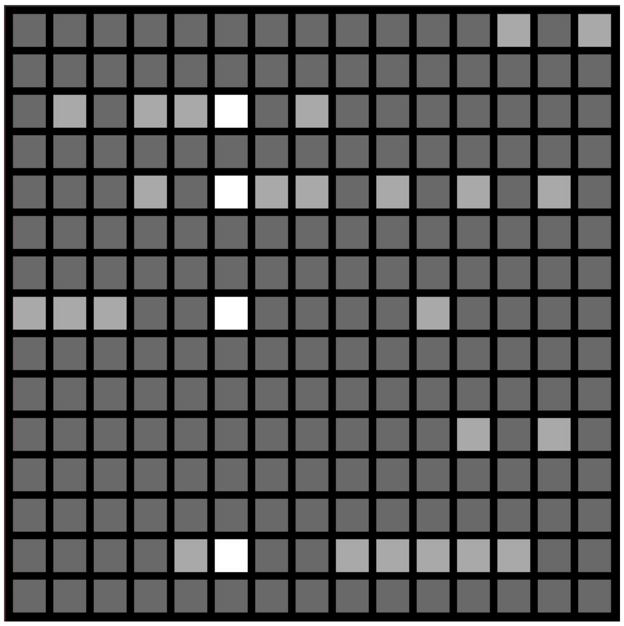
Sound Grid

Mini Project 4: Interactive Programming
Software Design Spring 2016
Gaby Clarke and Kristyn Walker

Project Overview
We created a sound grid, which is a matrix of blocks that displays notes in an audio loop visually. The user can click on blocks to turn them on, and activate the block. An internal timer sweeps through the matrix, so that every half second, one column of activated block are played. Once the last column is played, the loop repeats starting at the first column. This results in an addictive game of infinite musical rhythmic possibilities.

Results
Our sound grid is a square matrix of blocks where each block represents a sound. The blocks in each row play the same note, while the blocks in each column range from a high note to a low note down a musical scale.

The program has an internal metronome that keeps track of which column will play sound. It sweeps from the first column to the last column, then loops back to the first column. The metronome is set to increment every 150 milliseconds. On each increment, the next column of sound is played. The user activates a block by clicking it with their mouse. Once a block is activated, the block is illuminated light grey, and the sound of the block is played when its column is played. The sounds being played at a given time are illuminated white. At any time, the user can choose to activate or deactivate any of the blocks. The user can exit the program by pressing the escape key.

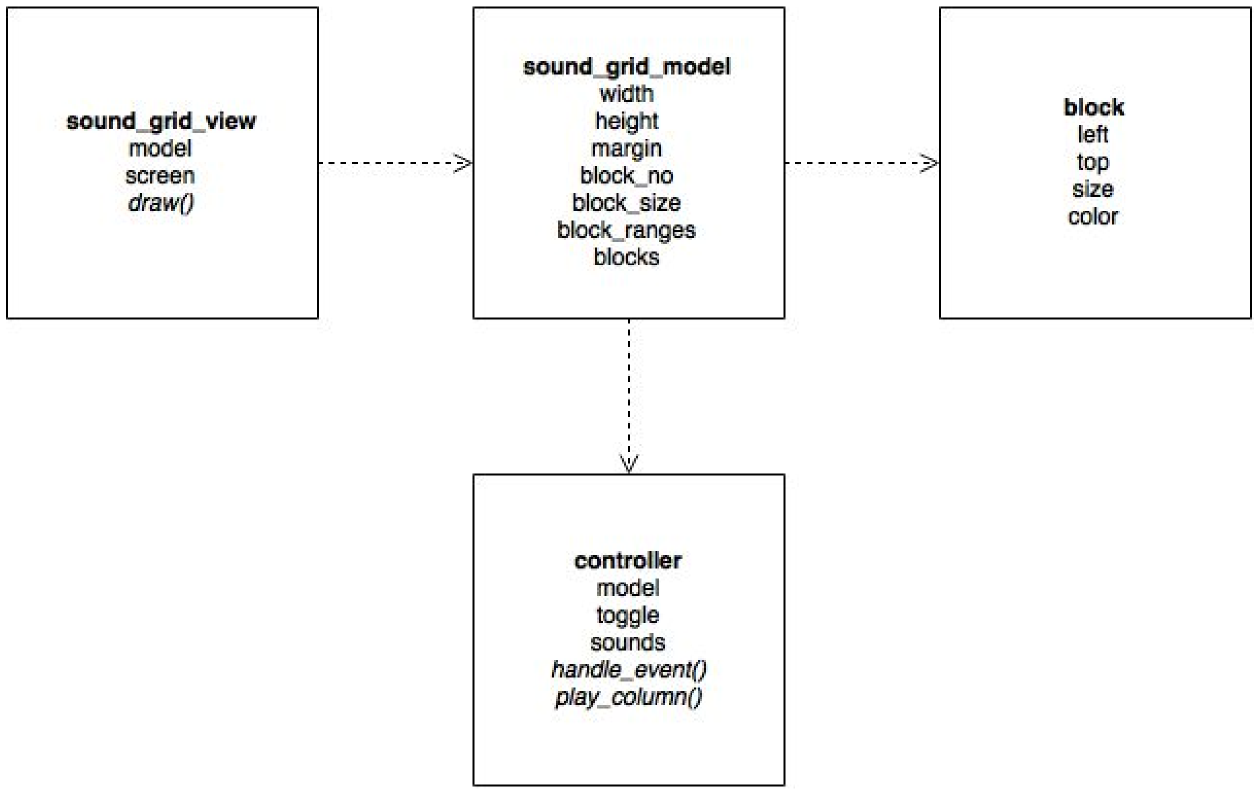



Implementation
We used `pygame` to implement our sound grid. The grid is a *n* x *n* square array of block objects (`pygame.Rect`) stored by column and drawn to a pygame display. Each block object is defined by its position, and the range encompassed by each block is stored in a corresponding array of block ranges so that it may be accessed upon the user's mouse press. Each block also corresponds to a toggle in an array which represents its status (unselected or selected), as defined by the user.

We used `pygame.time` to create our internal metronome by keeping track of the current time in increments of 150 milliseconds. Each increment is indexed in a modular system with base *n*, so that each index corresponds to a column of the grid.

The program can be implemented for any *n* as long as there are *n* sounds assigned.

Our block objects were initially stored as dictionary values with a corresponding block index key (the corresponding block ranges were mapped to a separate dictionary by index). We chose to instead store our block objects and their corresponding properties in 2D arrays because we could more easily access the columns of the matrix in an array than a dictionary. This is important because our sound grid is played by column, and the transition to arrays made the implementation of the sounds much simpler.



Reflection
We started the project off pair programming together and continued doing this throughout the entire project. Each time we met, we would switch off who typed the code. Partway through the project, we remembered that floobits is a thing. We shifted our process to pair programming on separate screens with more frequent shifting between roles of programmer and thinker/researcher/dictator. This worked pretty well, and allowed both of us to remain engaged in the project. It was really helpful for us work together, especially because we each thought about the process of adding in features differently. When we needed to add in the metronome, for example, we initially believed that it would be very challenging, and time intensive, but after we discussed the logic, it only took us a couple of minutes to implement. Another tool that worked for us was drawing things out on the whiteboard and talking through them. This really helped us to clarify our thoughts, reaffirm or correct our logic, and reach a shared understanding.

Regarding the project itself, we think we scoped our goals appropriately. We had initially thought that adding the sound component to the project would be easier than it was, and thus faced some frustrations when working on that, especially because we spent a fair amount of time searching for a good sound library to use and just ended up recording our own sounds. We also had a couple frustrations regarding the graphics, but we feel that we ended up making a program that is visually appealing, though it is not exactly what we initially intended.Overall we’re pretty satisfied with our end product.