Team NWA

Professor Clements

CPE 123

15 November 2013

Assignment #4: The Looper, Milestone #2

**Description:** For our project, Team NWA has come up with what we call “The Looper.” The function of the program is to allow the user to access an interface containing a virtual drum set and piano and record sounds on a repeating loop. As previously stated, the program contains various parts, such as the piano GUI or the looping code, that must work together at once in a single program that allows the user to create sounds and record over them on the repeating loop. The most important part of the program is the actual “looper.” This is what will allow the user to not only record their sounds over a time interval, but also be able to re-record over the recorded segment as many times as they please until they are pleased with their finished product. In addition to this feature, a key component to the final product is its appearance. This attribute is enhanced through the utilization of numerous GUIs. These include the main grid that contains the program, the drum set, the piano and the “looper” interface at the bottom of the screen. As the program is run, the loop will begin recording at the bottom of the screen as the user adds various notes and sounds, which will also appear when played on the “looper” interface. Once the user decides to restart the loop and record over the previous sounds or once the time interval is complete, the loop will start over and play whatever has been recorded while still allowing the user to add sounds. This design allows the user to hear a given group of sounds while still being able to interact with the “song”, per say, and add whatever is necessary until his or her masterpiece is complete. After complete, we are hoping to develop the code that will add a feature allowing the user to save his or her work for fun or future listening. This would be done by creating either a drop down menu or a save button anywhere on the main interface. Some of the interesting, more minor details regarding the project are things involving GUIs and how the program interacts with the user. This is done by implementing code that allows for a more visually appealing final product. For example, when the user hits a note on the piano, that key will be highlighted until it is released and the corresponding note will be added to the loop at the bottom. Also, when a drum note is played there will be a slight indent made on the virtual drum set as if the drum were actually being hit.

**Prototype:** Currently, our project is about 30 percent done, although split into a few separate parts. We have the current main program, which has support for playing the entire basic drum and piano sounds, as well as a basic GUI. The GUI is functional only in the piano keys, which will change shades when the corresponding keyboard key is hit. This is done by replacing the image on key-press with the darker shade, then replacing it with the original on key-up. The piano keys go up a full octave currently, from C4 to C5. There is also a drum section of the GUI and a section for the loop indicator; however, neither of these are functioning at the moment. The second part of the project is very similar; the interface is nearly the same but the piano keys are incomplete and do not correspond to key hits or contain the last key. The only difference is that this version of the program contains much more sounds than simple drums and piano, such as interesting synth notes. All these sounds are played by checking which key was pressed, and choosing a corresponding sound to play. The third current section of this project is the looper. Currently it is a separate program with no GUI whatsoever. It is built to work with only one sound, but is built in such a way that scaling it up to multiple sounds should be relatively easy. However, this looper has some problems that must be solved before implementing it program wide. The main issue is that, rather than playing the sound just once, it continuously plays several times. The exact amount of time through which the sound loops is currently unknown, but it is probable that it is around one big-bang tick, given that the code works by checking if the current tick is equal to the tick at which the sound was initially ticked to loop, and then queuing that sound at the current time, which could potentially cause the sound to run for that full tick rather than stopping after hitting it once. The looper also currently supports multiple positions for one sound (although with the previously mentioned error still present) by storing the positions for the sound in a list, with new positions added through a function that adds a new value to a list. Overall, the program is partially working in three separate parts, and has a lot of work to go, but is showing solid progress for a 30 percent project.

**Testing:** Our team’s program will be a looper that plays both piano keys and drum tones once a specific key is pressed. For the 30% milestone, we had to receive feedback from other people after they tested and ran our “Looper.” We would either show them our program on one of our laptops or we would send instructions on how to run it themselves from their computer. In those instructions we would include every step from downloading DrRacket to installing ‘portaudio’ and ‘rsound’ and running the actual program. The process of having others’ test our program included giving background information, giving them a glimpse of the code by scrolling down and giving a brief explanation about each section of the code and what it was supposed to do. We spent roughly 15 minutes giving the background information and explaining the sections from the code for the program. The people who tested our program were mainly students; they were roommates from the members in our group as well as other members from other groups in the CPE 123 classes. The students who are not CPE, CS, or SE majors were both surprised and confused after having a look at the 500+ lines of code the program had. Once the explanations were done, it was time for the testers to actually run the program. We showed them how to run it and once the GUI was up and running they figured out what to do and they began making their own tunes. Some of the people who tested the program would try to recreate parts of their favorite songs and play them using our program. After they understood how the program worked, they started to press the keys at a faster pace and would hold down some keys as well. Most users tested our program for a good 10-15 minutes. Overall, most of our testers were impressed with our “Looper.”

**Evaluation:** One key part to this project is finding out what part of our desired code works well, and what part doesn’t work so well. There are many ways we came about doing this. First we simply evaluated it as a team mainly through testing and communicating about how our project is coming along and in what direction it appears to be heading. Another way we evaluated it is through the input of our peers. To do this, we made a list of feedback questions that we asked to various people in order to see what they thought about our project so far. These questions asked were: “Is the GUI visually appealing? If not, what could be improved?”; “Do you think there are enough different sounds or is it boring?”; “Should we add a saving feature to save the final loop?”; “Would you enjoy using this program?”; “Did you find any bugs when using the program?”, and “Other feedback?” These questions were asked this week to a wide variety of students; not only computer engineering students but also students from all majors in Sierra Madre Tower 5, in order to get responses from a broad background of potential program users. The responses were similar within each group of people. For the first question (reference above), the computer engineering student feedback mainly said they would use a more interesting font on the interface. Also, they liked the drums and piano GUIs, but highly suggested adding a drop-down menu with a save feature and possibly other sounds via a sound selector. The non-computer engineering students said they thought the program interface was really interesting and impressive. For the second question, the most common response was the possibility of adding a sound selector with different sounds and instruments. For the third question, everyone responded yes, a saving feature should definitely be added. For the fourth question, the common response was yes, it would definitely be entertaining at least for a little while. Regarding the fifth question, the only big bug we have found so far was a problem with the highlighting of the piano keys when played. Occasionally the key will become permanently highlighted. When analyzing this feedback, we mainly noticed the common response of a possibility of more sounds, as well as the necessity for a saving feature. We have started to come up with more possible sounds for the loop, but have not yet started to implement a code for saving the users progress or final product.

**Refinement:** Given all this testing and feedback, it is important for us now to use this to our advantage and better our program. After our first running code was created, there was a problem involving the highlighting feature of the piano, which will be fixed to make sure the highlighting feature does not become locked onto the interface. Also, because it would be pointless to spend all your time working on a “song” on “The Looper” and not be able to save it, it is important that we work to add a saving feature to the program. Right now we have multiple parts of our code being developed in different parts, as it should be to avoid a giant mess. Working our way towards the next milestone and our final product, it is important that we soon start to see how these codes can interact and work together without any bugs to create the desired final program. Less important things that may or may not need to be improved are more personal choices, such as a change in font or color of the interface. One thing that was also brought up often in the feedback responses was the possibility of adding more sounds to the program. We have come up with additional sounds that may be added to the program, but are not yet sure if a GUI will be made to accompany the additional sounds. This may require too much of a drastic change to our original prototype, but we definitely would like to add more sounds to create a wider variety of sounds, thus making the program more interesting.