B. Weeks, T. Franklin, N. Dara, D. Grove

Dr. Clements

CPE 123, MWF 10:10-12:00PM

14 November 2014

Infiniano: The Prototype

*(NOTE: For clarity, the on-screen keyboard will be referred to as “the piano”, while the tangible, typed-on keyboard that takes user input will be referred to as “the keyboard” or “the user’s keyboard” in the writings below)*

**Description**

*Briefly describe the idea behind your product, the look of the program, and its basic controls.*

For the final project, our team decided to develop a virtual piano program controlled by the keys on the keyboard. The desired goal was to make it simple for the user to use, but relatively complex in terms of the extent of options the user has available if they choose. At the core of the program is a set of 24 piano keys (14 white, 10 black) spanning two octaves. The user can play each of these 24 keys by using the corresponding keyboard letter mapped to the piano key. Additional options available to the user are selecting one of eight different instruments (instead of the default piano), adjusting the volume of the program, playing demo songs to help illustrate the use of the piano, moving to a set of lower or higher octaves, and recording their work over their own melody or a premade beat. The idea is that this program can be used for a purpose as simple as sounding out a melody or as complex as creating a multilayered recording as a song demo.

The originally intended final design of the program is to be with the set of piano keys centrally located on-screen taking up approximately one-half to two-thirds of the window built by Racket. above the piano is a volume slider (scaled from 0 to 100), that allows the user to scale how loud notes will be played back. A playback button at the top left of the screen can be selected for recording options or to play a demo song. In the bottom left of the screen is a set of eight radio buttons that allows the user to choose their preferred instrument. In the bottom right region below the piano is a display which shows the octave set as a range of the midi note numbers currently displayed on the piano. Changing this is done with the up or down arrow key on the user’s keyboard. Additionally, an instructions page, selected by the left or right key on the keyboard, can be brought up to provide more clear information on how to use the program.

**Prototype**

*Describe what features and rules you have implemented in your prototype. You are required to minimally implement 30% of your final program.*

Our prototype incorporates most of the graphical interface as well as it’s functions; complete with the piano itself, a working volume slider, instrument selectors, and an octave toggler. The user can play the piano by using the corresponding key inputs. Additionally, they can play higher or lower notes by toggling the working octave selector. There is also support for playing these notes on a trumpet, pan flute, alto sax, harp, music box, string instruments, or synth pad. When a user selects their preference notes will be played in the style of the user’s chosen instrument. When a key is played, it is highlighted yellow on the keyboard until another key is played. There is not yet support for demoed songs or recording tracks, but it is being implemented.

**Testing**

*Describe the process the users went through in testing your program. How much time did you spend explaining how it worked? How long did they use it? Who tested your program?*

To test the program, we took a convenience sample and asked peers around us to use the program. After opening the program, we simply handed it to them. At first, we didn’t say anything and simply let them try to figure out the program by themselves in order to see how self-explanatory our program is. After a few seconds, when we first saw them struggle with a particular feature, we explained that piece of the program individually so that they could explore it to the full extent. Once they looked tired of using one feature, we directed them to another feature so they could witness the thus far full potential of our program. The current features we have include being able to play keys, changing octaves allowing for both low and high pitch notes, changing the instrument entirely, and the ability to change the volume of the sounds produced. Generally speaking, it didn’t take a vast amount of time to explain exactly how the program worked, partially due to the fact that our program is still incomplete and requires work to be done. Also, given that our program, a virtual piano, was relatively easy to understand, the user didn’t need a lengthy explanation. Our users enjoyed using the program for a few minutes, basically until they could explore all the features. Testers of our program include Jonathan Cheruvelil, Matt Doerner, Chris Berridge, and others, all first-years attending Cal Poly San Luis Obispo. From the efficient testing of the program, we gathered necessary feedback in order to maximize the potential of our program and extend its features.

**Evaluation**

In order to collect feedback from the testers we allowed a period in which they could tell us what they thought of the program and its features followed by a series of questions. A few of the major questions we asked are as follows. “What did you think of the user interface?“”What did you think of the idea of the program itself?” “If there were things you could change about your experience with the program, what would they be?” “What would be some new features that would improve our program as a whole?” In addition to these questions, we asked followups as well to understand exactly what they thought of the program and what they would’ve liked to see. We collected the information as we asked them the questions and received their feedback. After receiving the bulk of our feedback, we organized it based on the most prevalent feedback. One of which was the demand for an instructions page. Users tried to click on the keys using the mouse rather than press the keys. Even after we explained that the keyboard toggles the piano, they struggled to decipher which button corresponds to which key on the piano. Furthermore, our users did not like the fact that the piano keys stayed highlighted despite the fact that the sound being played was finished. Another strong feedback we received from the class is that our program lacks the capability to play 2 keys at the same time. Moreover, we received complaints about the interface being bland. The fonts weren’t elegant and the colors did not at all match well. One of our users noticed that by holding down a key, the piano replays the sound rapidly rather than playing the same song for a long time. Also, the midi-number explanation at the bottom-right served no use to the user. A repeated concern from our testing included the fact that our volume slider was redundantly large. It took up a large space and since it was just a volume feature, there is no need for it to be the center of attention. Lastly, the keys used to toggle the piano were disorganized. The users felt it didn’t make sense for certain keys on the keyboard to correspond to certain keys on the piano. Definitely, this feedback will be crucial in developing a stellar final project.

**Refinement**

Upon gathering feedback about our program, our team discussed which issues and enhancements we would most like to work into our finished product. Our thinking was to prioritize the simpler tasks with obvious solutions as these will allow us to build momentum as we fix issues, and many times smaller fixes must be made in order for larger enhancements to work. By combining tester feedback with some of the enhancements we’d originally planned, our task list is as follows:

1. *Fix piano highlight graphics* -- currently, when a key is played, the key will remain highlighted until another key is pressed. This is not how the program should be designed, and our first priority to improve the program’s response when a user presses a key; the key should stay highlighted when it is pressed and return to white when it is not. This will be fixed by including a conditional statement in our keyHandler function which tells the program to return a piano key to original color when the corresponding keyboard key is released.
2. *Enable key overlap* -- the program does not allow the user to play two keys at the exact same time. In turn the program will not highlight multiple keys when they are playing, and instead only selects the one that was most recently pressed. This cause-and-effect issue should be a relatively simple fix. At present, Infiniano only stores the midi number of the current note being played in the world-state. If we were to implement a list of the notes currently being played in the world-state of Infiniano, the program could play chords at the exact same time over top of one another. Additionally, multiple keys could be selected because the program would no longer interpret the world state as simply one key playing at a time.
3. *Make instructions more clear* -- a very common critique of our prototype was that it was not very obvious which keyboard letters were mapped to which on-screen keys, and changing octaves was unclear when these octaves were labeled by midi-note numbers. A suggestion was made that these keys be labeled, which we intend to include in our upcoming fixes. We also plan to rename the octave labels to “Octave 1”, “Octave 2”, and   
   “Octave 3”. Additionally, the group developed an idea of including an instruction page that the user could select when meeting to discuss solutions. On this page would be information about changing octaves, key mappings, and instructions on how to record a melody, once it is implemented.
4. *Beautify the interface* -- in order to make interface improvements as suggested by our test group, we intend to change the background and font colors, as well as create a smaller, more compact, vertical volume slider. We would like to add an image of a galaxy to supplement the colors of the rest of the program as well as to illustrate the overarching theme of our particular project: “infinite possibility through music.” Some of the graphic layout will need to be altered in order to modify the volume slider so that it takes up less space and is oriented vertically, and some accuracy will be lost as a smaller scale is less precise. The code allowing the slider to work will require only some minor position tweaks to line up with the new graphical position of the tool.
5. *Allow notes to be extended* -- to simulate a real piano, we would like a note to be extended when a key is held instead of looped over and over until the key is no longer being pressed. At the moment of this writing, we were not able to think of a solution that would fix this problem entirely. However, by spending more time working together to review Racket’s documentation and consider solutions, we expect to have this implemented in our completed program.Most likely, this will have to be fixed through a key-released conditional (play until key released) much like the piano key highlight issue. At the time of this writing, the program instead plays the note while the key is being pressed.

There are a few other enhancements we would like to add to the finished product. In addition to the refinements, we also have brainstormed entirely new additions and features that we wish to add to the program. Our users suggested having preset beats in which they could play over to form complex music. We also wish to have a metronome feature. Our program could have a main menu with multiple modes. For example, a mode with preset songs and the use of highlighted keys to teach the user how to play a certain song. Also, another mode could allow the user to record himself. In order to do this, we could have a main menu page and drastically improve and beautify the user interface. With this wish-list of features, Infiniano will truly provide the ultimate user experience.