Keyboard Music

Operation Manual

Gary Shigemoto
Brandon Stark

Music 147 / CompSci 190 / EECS195 Ace 277
Computer Audio and Music Programming
Final Project Documentation

Overview Description of Patch

The purpose of Keyboard Music is to allow the user to use a regular computer keyboard to create small simple musical rhythms and melodies. Keyboard Music is more than just a direct mapping between the keys on a computer keyboard and a musical keyboard.

A keyboard is primarily used to type words and sentences, but in this patch, it allows it to create small simple musical melodies based on the typical use of typing. One of the advantages of this patch is that the user is not expected to have any prior knowledge of composing or playing music. It allows non-music users to play a short melody on a keyboard that they are more familiar with using and takes into consideration the most frequent typing patterns such as starting sentences with upper case letters and frequent use of the space bar.

Creating a more complex sound is simple with the addition of a second lower octave instrument. Typing upper case letters by holding down the shift key selects the seconds instrument as well as sets a key signature so that all other lower case notes fit within the major scale of the upper case letter. The attack and length of the notes played are directly related to the speed of the typist. The faster the notes are typed, the shorter the duration and the higher the velocity.

Keyboard Music features six preset instrument sets:

1. **Tribal**
   Taiko Drums and FX7 (echoes) with Synth Drum.
2. **Techno**
   Agogo and Lead 1 (square) with Synth Drum.
3. **Orchestral**
   String Ensemble 1 and Tremolo Strings with Timpani.
4. **Piano**
   Piano and Piano with Timpani.
5. **Music Box**
   Music Box and Glockenspiel with Tubular Bells.
6. **Guitar**
   Electric Guitar (Jazz) and Electric Bass (pick) with
   Slap Bass 1.

Some of the presets such as Music Box encourage the user to type slower while others presets such as Techno encourage the user to type faster.

Examples

The user is free to explore the various possibilities of creating various little melodies on Keyboard Music but a good place to start might be with:

Sentences:
Hello
My Name is _____
Alligator
Cat in the Hat
Typing

Major Scale:
Aacefhjln

Simple Songs:
Mary Had A Little Lamb:
Akhghkkkhhhgkkkhghkkkkhhkhg

Explanation of Programming

Keyboard Music

The Keyboard Music patch starts with the keyDifferentialTimer sub-patch which outputs an ASCII integer value and the time in between key strokes. The integer value is divided by 1.0625 to represent a full 4 octave range. There are a total of 52 different letter keys that can be pressed, but only 48 pitches. The next step is to determine whether or not the shift key was held down while the key was pressed. Uppercase ASCII characters have an ASCII integer value of less than 91 after being scaled. If the note was lowercase, it is sent directly to the keySig patcher. If the note was not lower case, and was not the space bar, indicated by the ASCII integer value 30, then the note is shifted up by 6 to account for the characters in between Z and a and then inputted into the keySig sub-patch as well as simultaneously being converted into a note.

The makenote object takes in the integer value for the note, note duration and note velocity. The time in between keystrokes is used to calculate the note duration and the note velocity. Since frequently a user will type in short bursts and then pause for several seconds, the time in between strokes is evaluated in a logarithmic function. This ensures that a pause of 3 seconds will be only slightly different from a pause of 10 seconds, while still maintaining a significant difference between a pause of half a second and a pause of 1 second. For the note duration, the result of the logarithmic function is multiplied by 300 for lower case letters and multiplied by 1200 for upper case letters. This allows the user to use the upper case notes as a way of creating a simple chord. The note velocity calculation is more complex. The faster the user types, the faster the velocity should be.

So the inverse of the logarithmic function is taken by scaling down the result to be between 0 and 1 and then multiplying it by the range of the note velocity which is 255. The max value, 255 then has the resulting value subtracted from it. To prevent this result from going out of the range of the note velocity, a combination of an if statement and a switching gate is used. In this set up, the note velocity cannot exceed 255 and will not be below 100. At below approximately 100, the note velocity has a detrimental affect on the note played, especially with certain MIDI instruments.

The instruments that will be playing the notes are selected by a bang. Each condition, upper case letter, lower case letter and space bar have their own instrument that are selected and sent to the pgmout object each time they are pressed. There is also a different channel for each note, to allow that multiple notes can be played at the same time on different instruments.

For visual effects and user interface, a preset object containing six different instrument sets was included. This allows the user to easily choose a different sound without needing to know the MIDI instrument numbers. The instrument selector number boxes are preset to allow a more experienced user to select any MIDI instrument available. A keyboard is also included so that the user can see which note corresponds to letter.

KeyDifferentialTimer Sub-Patch

This subpatch is responsible for getting information on the keys typed. It captures what key was pressed on the computer keyboard and outputs the ASCII integer value and an ASCII character along with the delay between the current key types and the previous key types. When a key is pressed, one of two timers start. The next key pressed will stop that timer and simultaneously start the second timer. This alternating of timers accurately times milleseconds in between keystrokes. The sub-patch also has an inlet that takes a bang to stop both timers.

keySig Sub-Patch

The keySig Sub-Patch is responsible for adjusting the pitch of the regularly typed key to match the major scale of the key that was pressed while holding down the shift key. This prevents a lot of the dissonance that would be caused by typing two letters close together alphabetically. The sub-patch takes in the last key that was pressed while the shift key was held down and the current key that has been pressed. The shifted note is subtracted from the current note to set the octave. The result is then divided by 12 and the remainder is sent to a select object. If the remainder corresponds to a note that is not on the major scale, such as one half step above the root, the sub-patch adjusts it by adding an additional half step to the note, which makes it so it is on the major scale. The adjusted note is sent out through one outlet and a bang is sent out through the other. If no adjustment is needed, the note is still sent out through the outlet.

Possible Expansion

There is much that can be added to this patch to improve the functionality. Currently only two instruments and a single percussion sound is available, however that can be expanded to even more by including notes pressed while the Alt or Ctrl key is held down. This would allow a particularly skilled typist to control four different instruments at once. Another expansion on this project would to include better harmonic intelligence. While limiting notes to only major scale notes is useful for simple melodies, it is very limiting. Possible improvements might include ways of selecting a different mode of a major scale, or using minor or jazz scales. Including a way of creating more complex chord structures such as a dominant 5 or flat 7 would also beneficial. Currently, the only notes that are played are the notes that the user presses, however, by adding in a more complex chord intelligence system, chords or even perhaps a walking base-line could be added. The percussion sound also could be improved by including more special characters such as a period, comma or the enter key to allow for a more complex melody.

Bugs and Errors

There is occasionally a bug in which the switching gate in the KeyDifferentialTimer becomes out of synch with the timer that is currently running. This is solved by restarting the patch.

A significant problem that was encountered was the inability to use MIDI channel 10 to access the specialized percussion sounds.

It was very unfortunate that this patch did not work during the presentation. A couple adjustments have been made; however, none of the other computers this patch was tested on displayed any problems. The patch was originally developed on the computers in the Arts Media Center and also tested on our personal computers as well.

Tested Platforms and Versions:

Windows XP Professional Sp2a
- Max/MSP 4.5.7
- Max/MSP 4.6.2 Demo

Apple OSX 10.4
- Max/MSP 4.3