KeyKit − Musical Fun with Windows, Tasks, and Objects

Tutorial for Version 6.5a

Tim Thompson
AT&T
San Jose, California
tjt@nosuch.com

Introduction

KeyKit is a programming language and graphical interface for manipulating and generating music. This tutorial covers the basics of using its multi-window graphic interface, which is completely implemented by a user-accessible library of code written in the KeyKit language. A detailed language reference manual is contained in another document, but you don't have to read it right away since the graphic interface is complete enough to use without having to understand the underlying language. User-extensibility is a primary motivation behind the system, however, so if you become an active user, you should at least peruse some of the code in the user-defined library to see if you might be interested in adding features of your own to the system.

KeyKit is supported on a variety of computers. You should obtain installation instructions when you receive the KeyKit executables. The most widely accessible version runs under Windows 95 and Windows NT, so some details in this tutorial will be tailored to that version. In general, though, the user interface acts identically on all systems, including the behavior of pop-up menus and window manipulation.

An Initial Test

Start KeyKit (for example, under Windows, double-click the keykit icon in the Program Manager). A large white window should appear, and after a few seconds you should see a box at the bottom containing a `key>` prompt. This box is the Console, and contains an interpreter that will read and execute KeyKit statements. Type `print(44)` and it should print `44` followed by another `key>` prompt. Type and you should hear that 5-note phrase played on your MIDI output device. Press a few notes on your MIDI input device, and then type `print(sizeof(Recorded)).` You should see some number other than 0, and if you play more notes and type the same statement, the number should increase. You've now verified that MIDI input and output are working.

The default behaviour of KeyKit is to echo all MIDI input directly to MIDI output. This is appropriate when you have your keyboard controller plugged directly into the MIDI input of your PC, and your MIDI output directly drives your synths (or your soundcard). If you have your MIDI routing set up so that your controller is already sending its output to your synths, then your synths will be getting two copies of each note (which may not be immediately obvious except for a slight chorus effect in the sound). To disable KeyKit's echoing of MIDI input, you should execute `Merge=0` , by typing it in the Console window. You can make this change permanent by modifying the `c:\key\liblocal\prerc.k` file.

The **Merge** variable demonstrates the way in which many of the lower-level features (ones built into the language itself, rather than built out of the language in the user-defined library) are controlled by the values of special variables. The complete list of these variables can be found in the **keyvar(5)** manual page.

Menu Operation

Click the left mouse button anywhere in the background of the KeyKit display (i.e. anywhere except over the Console window). A pop-up menu, called the **main menu** from now on, should appear and remain visible after you

release the mouse button. The **Move** item in this menu is used to move windows. Click the left mouse button on **Move.** The menu will disappear and the mouse cursor should change to a cross, indicating that you are expected to now point to and drag a window. Click and hold the left mouse button down anywhere in the Console window (the box at the bottom). Drag the mouse, and the window will be moved.

Bring up the main menu again. The skinny horizontal strip at the top of the menu is a handle that can be used to move the menu. Press and hold the mouse button in this strip, and drag the menu around. The X in the upper-right corner of the menu can be used to erase the menu when you no longer need it. Note that if you just pop-up the main menu and select an item (like **Move**), the pop-up menu will go away immediately. If, instead, you first drag the menu around, the menu will be considered more permanent (posted), and will not disappear when you invoke an item. Hence the need for the X in the corner to erase it after it has been posted.

Menu items that have a −> in their names are nested sub-menus. When these items are selected, a sub-menu will be displayed, and it can be used (and moved) just like the main menu. As with the main menu, if you move a sub-menu it becomes posted and will not be erased until you explicitly remove it. This feature is intended to be used a **lot** - pulling off commonly used sub-menus and placing them on the screen (in locations of your own choosing) makes them easy to invoke.

Take a look at the **Tools2−>** sub-menu. Drag it away from the main menu so that it becomes posted. The left side of the menu shows a scroll bar, which you can use to scroll through all the items in the menu. (The width of the scroll bar can be controlled, if it is too small or large, by changing the Menuscrollwidth variable.)

The **Resize** item in the main menu lets you resize a window. Since menus are windows like any other window in KeyKit, you can resize them. Invoke **Resize** from the main menu, and point to the Tools menu that you've posted. The mouse cursor should then change to a "sweep" icon, and you can sweep (click and drag) the area you want the menu to occupy. In this way, you can make the menu as large or as small as you want. Try resizing the Console window for practice.

There is a special feature (that you may have already accidentally discovered) available for convenience when you are sweeping the area for resizing a window. If the area you sweep is extremely small (e.g. if you just click and release without dragging at all), KeyKit will make the window as large as possible without overlapping another window. This technique also works when you are sweeping the area for a new window.

Even more useful than pulling off sub-menus, pull-off buttons are a fairly unique feature of the KeyKit user interface. Bring up the main menu and then the **Misc−>** menu. The **AllNotesOff** item in this menu will send an all-notes-off MIDI message, which is useful when you have hanging notes (for whatever reason). Press the mouse button down in the **AllNotesOff** item, but don't release it. Drag the mouse button off the right edge of the menu, and keep dragging it right until it is an inch or two off the right edge. At some point, you will see an outline of a button appear, and you can drag this outline wherever you want to. Releasing the mouse button will reveal that you have just pulled the **AllNotesOff** item out of the menu and made it a separate button. This button is a window like any other, and can be moved and resized.

The pull-off button feature is meant to be used a **lot.** Any time you traverse a hierarchical menu and are about to invoke an item, at the instant you invoke it you can make the decision - do you want to invoke the item only once, or will you be invoking it multiple times in the near future? If you will need it more than once, you merely drag it out of the menu (no matter how deeply nested the menu) and place the button in a convenient place on the screen. That operation then becomes a one-click operation from then on. The button can be deleted when it no longer serves a purpose, using the Delete item in the main menu that is used to delete any window.

Tools

The items in the **Tools** menus allow you to create new windows containing tools. After invoking an item from a

Tools menu, the mouse cursor will change to the sweep icon, showing that you are then expected to sweep the area for the tool. As mentioned above, if you merely click the mouse rather than sweep any area, the tool window will be made as large as possible without overlapping any other windows. If you accidentally create a too-large tool, just use **Resize** in the main menu to resize it.

Select the **Mouse Matrix** tool. Sweep out an area (approximately square), and the result will contain a grid. Click and drag the mouse button within the grid, and you should hear a series of chords played via MIDI output.

Helpful information about individual tools can be obtained by looking at the **tools** reference document. In addition to being available in hardcopy form (see the **doc** directory for various formats such as Postscript), it is available online. On a Windows system, it is available as a standard Help document. You can access it and several other documents by pulling down the Windows menu (the one containing the "-") in the upper-left corner of the KeyKit window. You can also access the tools documentation by pressing F1 while the mouse is over the tool on which you want information. Doing this, you will be taken directly to the page describing that tool. You can also invoke the **Window->Help** item (in the KeyKit menus) and explicitly point to the window on which you want help. It may be a good idea to tear off this **Help** item and place it on the screen, so you can request help on a tool at any time, and so you are reminded that the help is available.

Now, back to experimenting with tools. Select the **Volume** tool. Sweep out an area (tall and thin), and you'll find a slider. When you drag the mouse in this slider, it will send volume messages on all 16 channels. Ignore the buttons at the bottom of the slider, for the moment. Move the Volume slider, then go back and make some more music with Mouse Matrix - you should be able to control the volume of the output, assuming your synth pays attention to the controller messages that the Volume slider sends.

Select the **Tempo** item. Sweep out an area (tall and thin), and you'll find a slider. When you drag the mouse in this slider, it will control the current tempo of whatever music is being played back (which we'll be doing in a moment).

As you can probably tell by now, using KeyKit often results in a rather chaotic-looking screen, since everything you do results in a separate window that is manually placed and resized. There is a way to arrange windows more neatly. First, find the **Misc->Windows->Arrange** menu. Pull off this menu so that you can repeatedly invoke its items. The **Horizontal** item in this menu allows you to select several windows and have them arranged horizontally in a specified area. The two sliders (Volume and Tempo) are a good candidate for this rearrangement. Invoke the **Horizontal** item, and then click the mouse once in each of the windows you want to select (in this case, Volume and Tempo). To indicate that you have selected the last window, click the mouse a second time in the last window. The mouse cursor will now change to the sweep icon, and you should sweep out the area in which you want the windows to be arranged.

Before we start playing any real music, a metronome would be useful. In the **Misc** menu, you'll find a **Toggle Met.** Selecting it will turn on a metronome, which by default is MIDI note 40 on channel 10 - a snare drum according to the General MIDI standard. You can change the note used for the metronome by changing the **Met** variable. For example, executing the statement `Met='p41c8'` in the Console window would set the Metronome to MIDI note 41 on channel 8. The default value of `Met` is set in the **keyrc** function, which is defined in `c:\key\lib\bootutil.k`. If you want to permanently change the value of Met, you should edit the file `c:\key\liblocal\postrc.k` and set it there. The **postrc** function gets executed after **keyrc,** so it will override the default.

You will probably want to toggle the metronome on and off fairly often, so the **Toggle Met** item is definitely something that you'll want to pull off and make into a button. If you don't know how to do that by now, this is a good time to learn. Pulling off a menu and pulling off a button are not the same thing - read the section above on Menu Operation for more details.

The **Riff** tool is the simplest way of playing a Standard MIDI File. Select **Riff** tool, and sweep its window. The

More button is an example of a menu button embedded in a tool - when you press it, a drop-down menu will appear. This menu works like any other menu, which means you can pull the entire menu off, and you can pull off the individual items as buttons. Select the **Load from File** item from this menu, and a standard Windows dialog box for finding and selecting a file will appear. You will find some sample music files in the `c:\key\music` directory - select `bachinv1.mid`. The **Riff** tool will show the music in piano-roll form. Click the mouse anywhere in the piano-roll window, and it will begin playing. Click the mouse again and it will stop.

Start the music playing, and then use the **Tempo** tool to adjust the speed. Note that you can do anything at the same time as the music is playing, including bringing up another tool. The only time that the operation of KeyKit is interrupted is when you bring up the Windows file selection dialog box (a problem that may be fixed someday, if it proves to annoying). Other than that one exception, the operation of all tools in KeyKit is continuous and simultaneous - a critically important feature that is made easy by the fact that the KeyKit language is multi-tasking.

KeyKit has support for object-oriented programming, and the implementation of the tools takes advantage of this. Most of the object-oriented aspects are hidden in the user interface implementation, but the **Bang** tool is one example that reveals object-orientation at the user level. The **Bang** tool allows you to send an arbitrary message to any number of tools. First, create a **Bang** tool. Make sure you have a **Riff** tool elsewhere on the screen. Now, click and hold the mouse down within the **Add** button of the **Bang** tool, and drag the mouse away - you should see a line stretching with it. Drag the line's endpoint so that it lies anywhere within the **Riff** tool, and release the mouse button. You've now attached the **Bang** tool to the **Riff** tool - to check, click the mouse on the **Add** button, and the **Bang** tool will show it's current connections (in this case, the one you just made). You can make any number of connections from the **Bang** tool to other tools. Now, press the **Bang** button in the upper-right of the **Bang** tool. This will send a "bang" message to all the attached tools. The **Riff** tool responds by starting to play its music. If you send another "bang" message, the music will stop. The **More** menu of the **Bang** tool allows you to change the type of message sent. More interestingly, the **Bang** tool can monitor MIDI input, and when it sees a particular MIDI note, trigger the sending of the message.

Pages

Once you have filled the display with tools, you can either start deleting them (with the **Delete** item in the main menu), or you can move to a new **page.** Each page is like a virtual screen, with its own set of tools. Each page also maintains its screen size (i.e. the size of the Windows window containing it). You can maintain as many pages as you like, and can save each to a file for later restoration. The conventional suffix for a page file is A file contains all of the information on the page, including any of the musical data contained in tools such as the **Riff** tool. A file is readable ASCII data. In fact, it merely contains KeyKit code that will reconstruct the page. Hence it is portable between different machines and machine architectures. It is also editable, which may come in handy if bugs or changes in KeyKit prevent it from being successfully loaded.

Select **Page->New.** You will see a blank screen. Add some tools to the screen. Now, use **Page->Switch->Page 1** to return to your first page. Likewise **Page->Switch->Page 2** will go back to your second page. You can use **Page->Label** to change the name of a page, which will affect the name you see in the **Page->Switch** menu. **Page->Write** lets you write the current page to a file, and **Page->Read** lets you read a page file.

The most convenient way to use pages is **Page->Snapshot.** When you select this, the current page will be written to a file whose name is of the form **snap#.kp,** where **#** is an integer. Each time you do a **Snapshot,** the integer will be incremented so that a new file is generated. So, if you're working on some music, and want to periodically save your effort, just pull off the **Snapshot** item and make it button, and just press that button every time you want to save your work.

The Windows installation of keykit should "assocatiate" the file suffix with the KeyKit executable. That way you can just double-click on a in the File Manager and KeyKit will automatically be invoked and read that file.

Group Tool

The **Group** tool is very much like a standard multi-track sequencer. It allows flexible editing of music, and is the most important (and largest) tool provided in the user interface of KeyKit.

Create a **Group** tool. Use the **File** menu button in its upper-left corner, and invoke **Read->Standard MIDI File** Using the dialog box that appears, select `c:\key\music\prelude.mid`. You should see a piano-roll representation of that file. You should also see a box with some probably-unreadable text, at the beginning of the piano-roll display. That is a Tempo message - other Standard MIDI File messages (such as Timesig) are displayed similarly.

The Group tool maintains a set of tracks. The merged contents of all tracks is always displayed in the track named **Merged.** Using the **View** menu button, select **Tracks->All.** You will then see the Merged track along with the 3 individual tracks that are contained in the `prelude.mid` file. At the upper-left corner of each track (including the merged track) is a menu button, and the menu it reveals will be called the **track menu.** The items in a track menu apply only to the corresponding track. For example, the **Unshow** item in the track menu will cause that track to be removed from the display (though it still exists, and its music will still be shown in the Merged track. The **Showonly** item will cause only that track to be display (in addition to the Merged track, which is always shown. If you want to go back to seeing only the Merged track, select **Showonly** in the track menu for the Merged track.

The text within the track menu buttons are the track names. You can change a track name by using the **Label** item in the track menu.

Within each track window, the 2 mouse buttons (left and right) are used for a variety of operations. Instead of being fixed, the meaning of each mouse button can be changed, and in practice you should expect to change their meanings often. The meanings of the mouse buttons are assigned with the 3 menu buttons in the upper-right corner of the Group tool. Each mouse button actually has three meanings - one when the shift key is pressed, one when the control key is pressed, and one when no keys are pressed. The keys that can modify the mouse button meanings are called "modifiers". The leftmost of the 3 menu buttons in the upper right corner of the Group tool controls the "modifier" that currently applies to the other two menu buttons - the modifier value can be **Normal, Shift,** or **Control.** When the modifier is set to **Shift,** then the other two menu buttons will control the meanings of the left and right mouse buttons when the Shift key is pressed. Likewise, when the modifier is set to **Control,** then the other two menu buttons control the meanings of the mouse buttons when the Control key is pressed. And when the modifier is set to **Normal,** then the other two menu buttons control the meanings of the mouse buttons when no key is pressed. The label of each button reveals the current meaning of that mouse button (for the current modifier setting). To change the operation of a button, you select the operation you want from the corresponding mouse button menu, and from then on that mouse button (combined with the appropriate modifier) will perform that operation.

The default operation for the left mouse button, when a Group tool is first created, is **Aud Sweep** (where **Aud** is short for **Audition**). This means that when you press and drag the left mouse button within a track window, you will be sweeping out the audition area. Try it, making sure that the mouse is within a track window when you start sweeping. While you sweep, you will see vertical bars that indicate the audition area. When you release, there is no visible indication of the current audition area (though someday there probably should be). The default operation for the right mouse button is **Aud Play** - to play the current audition area. So, if you use the left mouse button to sweep some part of the music, and then press the right mouse button, you should hear that music played via MIDI output, and you should see the notes flash as they play. To stop the playback, press the right mouse button again. This shows that **Aud Play** is a toggle - it starts and stops playback.

Now, try changing the meaning of the left mouse button. Use the middle menu button (of the 3 menu buttons in the upper right corner of the Group tool) and select **Pick->Sweep.** From then on, the left mouse button will sweep out the current **Pick.** The term Pick refers to the selected notes of the music that will be affected whenever an Edit operation is used. Try using the left mouse button to sweep an area of the music. After you sweep, you should see the

notes in that area turn red. The notes of the current Pick are always drawn in red.

The **View->In** menu item lets you zoom in to get a closer look at the music. After you select this item, the mouse cursor will change to a left-right arrow, meaning that you are then expected to sweep out an area within one of the tracks. Do this. You should see the display redrawn with that area of the track filling the display. Note that **View->In** changes the meaning of the left mouse button only temporarily - it then reverts to the normal operation (that you have specified). **View->Out** will return to viewing the entire piece of music.

Zoom in on part of the music, so that you can easily see individual notes. Pick a few notes (i.e. make sure the left mouse button is set to **Pick->Sweep,** and then sweep the notes). Now, select **Edit->Delete,** and the notes in the current Pick will be removed. Do it again - pick some more notes and delete them. Now, select **Edit->Undo.** You will see your last deletion undone. Select **Edit->Undo** again, and you will see your first deletion undone, so your music is back to its original state. The **Undo** operation can undo the last 32 editing operations, by default, and you can easily increase this number, subject to memory limitations.

The **Edit->Undo** item is one of the most common menu items to pull off and make into a real button. It's important to realize that when you do this, that **Undo** button will only affect the Group tool from which you pulled it - it will not affect any other Group tools that are on the screen. This is true of any of the buttons that you pull from menus in any given tool. There is currently no visible or other indication of the relationship between pulled-off buttons and the tools they control, so it's obviously a good idea to position the buttons in ways that make it obvious as to which tool they belong.

More Tools

Below are descriptions for the rest of the current set of tools - the descriptions aren't comprehensive, but should give you a sense of what each tool does and how it is used. These descriptions are meant to be used while you experiment with the interface - they may be a bit cryptic if you don't have the tool in front of you to play with.

The **Bang** tool is a way of sending arbitrary messages (also known as "methods" in the underlying implementation) to selected other tools. You use the **Add** button to add connections, stretching a line to the tool to which you want to send a message. A typical message might be "bang" or "on" or "stop". For example, sending a "bang" message to a **Riff** tool will begin playing the phrase in the riff tool. The **Bang** tool can be manually controlled by pressing its **Bang** button, but it can also be told to monitor MIDI input for a selected note, and will trigger the sending of the message. The **More->Load Recorded** menu item establishes the MIDI note that will trigger it, and the **On** button of the Bang tool is used to turn on the monitoring of MIDI input for this note.

The **Blocks** tool is most typically used for piecing together drum patterns. It introduces the concept of a file (standing for **KeyKit Collection**), which contains a set of named phrases. In the case of a drum pattern collection, each phrase is a complete drum pattern. The phrases need not be of the same length. You can use the **More->Read Collection** menu item to read a file. As an example, the file `drums1.kc` file is included in the standard **lib** directory of KeyKit - it contains a set of drum patterns that use the General MIDI standard for drum notes (on channel 10). After reading this collection, the menu buttons above each block of the **Blocks** tool will reveal, when you press them, the list of patterns. Select one of the patterns, and it will appear in the corresponding block. The default number of blocks is 4, but this label is actually a menu button that lets you control the total number of blocks. Clicking the mouse in any given block will start playing that block, and continue through to the last block on the right end. If you want the blocks to loop, you can turn on **Loop,** which is a toggle button. To stop playback, just press the mouse button again in any block.

The **Bounce** tool is a strange toy I built in an hour, making effective use of the object oriented implementation of the KeyKit user interface. Basically, it is 4 copies of the **Riff** tool surrounding a box. When you press **On,** a bouncing line will appear in the box. Each time the line hits a border of the box, the **Riff** on that side of the box will begin playing. When the line hits the same border again, the **Riff** will stop (if it's still playing). The four **Riff** tools within

Bounce operate identically to the standard **Riff** tool (because they **are** the standard **Riff** tool). It is best to fill the 4 phrases with either single chords or drum notes, to get started. The slider controls the speed of the bouncing line.

The **Chord Palette** is a way of playing chords - make sure you make its window large enough so that the labels are completely displayed rather than being truncated. Clicking the mouse in the cells of the matrix will play the corresponding chords. The chords are also (by default) put into the current value of **Snarf.** So, you can press a chord here and the go to some other tool to make use of it - for example, by using the **More->Load** from Snarf menu item in a **Riff** tool.

The **Comment** tool is merely a way of placing labels on the screen. After placing it, click the mouse on it - you will be given an opportunity to change its text. Very crude.

The **Console** tool allows you to have additional console areas in which you can type KeyKit statements and have them interpreted. When you have multiple consoles, error messages (and the output of print() statements, which you might embed in KeyKit code for debugging purposes) will appear in whichever console is "current". Clicking the left mouse button in a console window will make it the "current" one.

The **Echo** tool is used to echo MIDI input. Sliders control the echo time, transposition, volume decay, and number of echoes. The echos do not occur by default, though - you need to press the **On** button (which is a toggle) in order to activate the tool. You can have multiple **Echo** tools running simultaneously (as with any tool), which can be a lot more fun than a single echo.

The **GM Prog Map** tool lets you transmit Program Change commands on all 16 channels. You select patches according to the General MIDI standard. Each label is actually a menu button that reveals the patch list. When you make a change, the corresponding program change message is transmitted via MIDI output.

The **GM Control** tool lets you transmit controller messages such as Volume, Expressions, and Pan on all 16 channel. It also includes some of the General MIDI standardized controller values such as Reverb and Chorus depth.

The **Kboom** tool is a drum pattern editor that has taken on a life of its own - it has a lot of features that are a lot of fun to play with. The steps of the drum pattern are not restricted to drum notes - they can be arbitrary notes or even phrases. Use the **Read Drumkit** menu item to read the `lib\drums.kbm` file - it is an ASCII file that specifies a drum kit. You can interactively select the drums for each row in the **Kboom** tool, by using the labels which are menu buttons. Click the mouse within the cells of the grid in order to toggle the drums on and off at each step of the pattern. Instead of having each step be just on and off, you can use **More->Toggle Mode->Gradual** to turn on a mode in which each cell can be filled in gradually. If a cell is filled in 25% of the way, then that "hit" of the drum will only occur 25% (randomly) of the time. This can add variety to a pattern. Another way of adding variety is to create a second **Kboom** tool with a different number of steps, and playing the two patterns at the same time. You can also use the **Rec Note/Chord** item in the menu buttons on each drum row to grab the last-recorded MIDI input, which is then assigned as the sound for that row. This can all be done while the drum pattern is playing.

The **Markov Maker** tool lets you create music with markov-chain techniques. Use the **Orig** button to read in some piece of music. Then use the nested menus under **Orig->Set Sim** to initialize the Markov model of the music. You can experiment with the values, but for good initial results, try **Win 1b/2** and **Inc 1b/4.** The **Win** value is the "window" that is swept across the phrase, and the **Inc** value is the increment by which the window is moved across the phrase. (See the source code in `lib/setsim.k` for more details.) Now, use the **Sim->Make Sim->** menu to select the size of the "similar" phrase that you want to generate. Pressing the mouse button inside either phrase window will start (and stop) its playing. You can use **Sim->Snarf** to grab the generated phrase and then (for example) read it it into a **Group** tool for further editing. The **Markov Maker** tool is perhaps the easiest thing to impress people with. As an example, use `music/prelude.mid` as the music, and **Win 1b/2** and **Inc 1b/4** as the values, and generate a fairly long piece. It’s surprising how interesting the results are.

The **Parameters** tool lets you set various global parameters. For example, the **Sweepquant** and **Dragquant** parameters control the quantization of sweeping and dragging within the **Group** tool.

More Information

The **doc** directory contains:

tutorial	This tutorial.
tools	A reference manual for the individual tools in the user interface.
language	A reference manual for the KeyKit language.
hacking	Information on modifying and adding tools in the KeyKit user interface.
keyvar	Manual page for the built-in variables whose values can be modified to control KeyKit's operation.