Simulating media controller buttons like Play and Pause

devblogs.microsoft.com/oldnewthing/20140929-00

September 29, 2014

Raymond Chen

Today's Little Program simulates pressing the Play/Pause button on your fancy keyboard. This might be useful if you want to write a program that converts some other input (say, gesture detection) into media controller events.

One way of doing this is to take advantage of the `DefWindowProc` function, since the default behavior for the `WM_APPCOMMAND` message is to pass the message up the parent chain, and if it still can't find a handler, it hands the message to the shell for global processing.

Remember, don't fumble around. If you want to send a message to a window, then send a message to a window. Don't broadcast a message to every window in the system (resulting in mass chaos).

Take the scratch program and make this simple addition:

```
void OnChar(HWND hwnd, TCHAR ch, int cRepeat)
{
 if (ch == ' ') {
  SendMessage(hwnd, WM_APPCOMMAND, (WPARAM)hwnd,
      MAKELONG(0, FAPPCOMMAND_KEY | APPCOMMAND_MEDIA_PLAY_PAUSE));
 }
}
 HANDLE_MSG(hwnd, WM_CHAR, OnChar);
```

When you press the space bar in the scratch application, it pretends that you instead pressed the *Play/Pause* button on your fancy keyboard with no shift modifiers.

The scratch program doesn't do anything with the key, so it ends up falling through to `DefWindowProc`, which eventually hands the key to the shell and any other registered shell hooks. If you have a program like Windows Media Player which registers for shell events, it will see the notification and pause/resume playback.

Of course, this assumes that the program you want to talk to listens globally for the keypress. If you want to make the current foreground program respond as if you had pressed the *Play/Pause*, you can just inject the keypress.

```
int __cdecl main(int, char**)
{
 INPUT inputs[2] = {};
 inputs[0].type = INPUT_KEYBOARD;
 inputs[0].ki.wVk = VK_MEDIA_PLAY_PAUSE;
 inputs[0].ki.wScan = 0x22;
 inputs[0].ki.dwFlags = KEYEVENTF_EXTENDEDKEY;
 inputs[1].type = INPUT_KEYBOARD;
 inputs[1].ki.wVk = VK_MEDIA_PLAY_PAUSE;
 inputs[1].ki.wScan = 0x22;
 inputs[0].ki.dwFlags = KEYEVENTF_EXTENDEDKEY | KEYEVENTF_KEYUP;
 SendInput(2, inputs, sizeof(INPUT));
 return 0;
}
```

Note, however, that since we didn't do anything about the state of modifier keys, if the user happens to have the shift key down at the time you injected the message, the application is going to be told, "Hey, do your play/pause thing, and if you change behavior when the shift key is down, here's your chance."

But what did you expect from a Little Program?

Raymond Chen

Follow