OpenXTalk Don't Panic! Edition

Integrated Development Environment

Welcome to this first release of OpenXTalk DPE IDE!

As some may know, The OpenXTalk IDE is built upon the now-defunct open-source LiveCode Community Edition, which the Edinburgh based LiveCode Ltd. dropped all support for on Sept. 1st 2021.

When LiveCode Ltd. suddenly abandoned open-source I thought to myself that there were many more possibilities for an open-source xTalk interpreter which is available free to the general public then perhaps are suitable for capitalizing on or otherwise monetizing at this moment in time. Moreover I thought that perhaps the commercial interests and direction guiding development, the dual-licensing model compatibility requirement for adding to the IDE, a concentration on business or database users, and perhaps not so much interest in hobbyists that code for fun or as an open learning tool for educators or for young coders just starting to learn some programming concepts (and I don't think there is any other programming language that's as easy to learn or as psuedo-code like than xTalk is), these things may have actually negatively impacted on the possibilities for community users in some ways.

So I decided that I would dedicate my free-time (which is sometimes very little) exploring how the IDE and xTalk interpreter(s) are constructed and to work on this code-base (LC CE v9.6.3) to try to not only maintain its public availability, but also to improve upon it however I can along the way, fix some old annoying bugs, and try to complete some, in my opinion unfinished, promised 'stretch-goals' from multiple crowd-source fund-raising drives that the community had funded. This first release of OpenXTalk IDE is the result of over a year of working towards these things.

Changes

DeBranding / ReBranding

Firstly the IDE needed to be 'de-branded' so as to not 'confuse' any customers of LiveCode Ltd. now closed-source, commercial-only product line. Towards that end, all references to 'LiveCode' needed to be either removed (if they could be removed without rewriting too much of the IDE), edited, or a replaced. For that to happen many of the IDEs stacks and scripts needed editing.

The named OpenXTalk was chosen after brief, informal community polling on FaceBook. Wherever possible completely generic verbiage was used instead. For example a menu item such as 'Show LiveCode IDE Stacks in Lists' would change to 'Show IDE Stacks in Lists' rather than 'Show OpenXTalk IDE Stacks in Lists'.

IDE Executables and File Types

The IDE Applications executables were altered (although not yet re-compiled from C++ source). On all platforms the application and document icons where changed. Alternate file-extensions have been added, '.oxtstack' (GUI stack) and '.oxtscript' (aka script-only stack). On macOS the IDEs Info.plist, was edited, on Linux a new .desktop launcher file created, and on Windows the executables metadata have been edited to enable these new icons and filename extensions. Related IDE scripts for have also been edited to enable recognition of these new additional file types. For now the stack file-formats remain the same, with the single exception that in OXT IDE GUI stacks have new 'darkMode' property. An additional new file type is planned for a user interface markup and IDE script parser for the generation of graphical UIs stacks from UI files in a GitHub / version-control friendly plain-text format. This is meant as a compliment to the current plain-text script-only format for stacks without any UI.

There are two versions of the macOS binary, one that is signed with an OXT (Ad-Hoc) signature, and one with all code signing stripped from it. I recommend using the unsigned version, which will ask the user for permissions to access certain operating system APIs when they're first called by the IDE (administrator password is required). The signed version was mostly an experiment in sandboxing and adding entitlements to the IDE. For security reasons may want to sign any standalone app executables that you deploy with your own developer certificate.

Home stack and legacy licensing

The Home stack, which is like the IDEs 'mainStack', was edited quite a bit, and some new handlers were added. Excessive (in my opinion) start-up logging was removed, so the IDE may load a little faster now as there is less disk writes going on during the IDE start-up process. License checking and license upgrading related things have been removed because there is now only one licensing model now, and that is the General Public License (GPLv3 with some exceptions) license that was inherited from the LiveCode Community Edition. There will never be any closed source, commercial licensing for OpenXTalk IDE. If you need that sort of thing, DRM and such, then perhaps you should invest in a subscription to LiveCode Ltd.'s new IDE as a service model (but be warned that model may tether apps you build to their internet licensing server). The one exception is the initial license check and embedded account creation stack. If an account creation dialog shows up when you first install OXT IDE, you can simply dismiss this dialog, a dummy license file will be created and then the IDE should launch normally thereafter.

HyperCard stacks

If you're a long-time xTalk user like me you may have continued using HyperCard into the mid-2000s (until support for macOS 'Classic' was dropped in MacOS X 10.5 Leopard), long after development efforts from Apple had stagnated (and plans for integrating an HyperTalk interpreter into QuickTime 'Interactive' had been abandoned in favor of concentrating QT development on media streaming), then you may still have old stack projects around that you may want to take quick look at, extract some script handlers from, or update for the modern era. HyperCard stacks were (semi-)importable in previous (v6.x and lower) versions of the IDE, provided that the stacks did not rely too heavily on additions like XCMDs/XFCN and provide that the stack was 'compacted' when it was last saved in HyperCard. Since version 7 trying to import HC format stacks resulted in an IDE crash. The IDE crashing has been corrected on the macOS platform (the only platform that HyperCard supported, unless you include the ancient HC v1.2 16-bit port for the AppleIIGS platform). Now trying to open an HC stack in OpenXTalk IDE on macOS will either fail more gracefully (no crash) or, if the open-source application called 'HyperCard Preview' is found on the system, the stack(s) will be passed to that application for viewing (but not editing) of the HC stacks contents. This will allow you to copy HyperTalk scripts from old stacks, even bypassing HyperCards stack password protection, and it can read un-compacted stacks just fine as well. There is interest from retro HyperCard users, support may be expanded on to support importing again and to support other IDE platforms via porting Uli Klusters 'stackimport' command-line HC to JSON/PNG conversion utility.

Graphics

Since logo graphics where used in many places in the IDE, new graphics were needed as replacements. I thought it would be appropriate to pay homage to the original xTalk, Bill Atkinson's HyperCard, which was officially dropped by Apple roughly two decades ago (yet still has a fan-base as well as some new retro-users that may not have even been alive back when). I designed the OpenXTalk IDE logo/icon graphics intentionally and obviously based on some of the HyperCard graphics, but where possible in a modern vector graphics format so that they're resolution independent, can be used as a SVG path icons, or can be rendered to pixels (PNG, JPEG, etc.) at any given size (because 8K px screens exist now). You may find more HC inspired graphics sprinkled in here and there, such as the 'stack' icon used in the Project Browser. The plan for graphics going forward will be to replace as many pixel-based static images with SVG vector graphics, which should help to reduce the overall size of the IDE (currently a gigabyte disk footprint).

'DarkMode' Support

Since, in the state it was left to the community in, the IDE still lacked all but the most basic support on macOS for the increasingly popular 'darkMode' (I like it), and since I had already developed Extension Builder API hooks into macOS native darkmode, it was a good fit for visually altering the IDE in a significant way. Supporting darkMode on macOS can involve more then just changing your internal graphics, the operating system provides 'dynamic' colors that the OS can automatically adjust for higher contrast, window frames are toggled to/from light to dark, some windows can now have an effect where a blurred image of what is behind a window bleeds through (not yet fully implemented for OXT), and in most cases the foreground/text and background colors are automatically swapped. However, OXT on non-macOS platforms was lacking any 'darkMode' support (dark GUI themes), which is not available as any API at all on some platforms, the xTalk engine then delivers incorrect reporting (always returns 'light'). As a substitute, an xTalk script-based faux darkMode was created which alters the look of IDE stacks in memory. Additionally, many of the IDEs stack (GUI) files needed to be edited because black text on a dark background is not good for readability. If a stack objects or controls have color properties set they may need to be altered for the appropriate look. If no color properties are set on a card, stack, or controls then there may be no need for any editing because the properties colors should be inherited from the xTalk engine itself, which in turn should inherit the OS provided default properties for its current window manager theme. Stacks have a new 'DarkMode' property that can be set. Although this is not (yet) automated by the IDE in any way, you can already use this properly for your own 'open stack' handlers to check if the property is present and use its value for accordingly scripted (light/dark UI) graphical alterations to your stacks.

The StartCenter

The previous StartCenter stack has been replaced with a 'Stack Forwarder' stack that first looks for and if found loads any stack called StartCenter.oxtstack in the IDE users 'My OpenXtalk' folder located in the user documents directory. If no user StartCenter stack is found then the default StartCenter included with the IDE will be loaded. This allows for experienced users to completely customize their StartCenter with anything that they would want readily available at IDE startup, similar to the customization role that HyperCards Home Stack was often used for.

In contrast the IDEs default StartCenter contains introductory materials targeting users that are completely new to xTalk environments and may have no idea what a stack or a card is in the context of xTalk. The default StartCenter can also launch the various guided lessons or demonstrations (such as standard "Hello World"). Documentation for the API used to build these guided lessons is now included in the guides section of the Dictionary stack.

Main Menubar and its attached Toolbar

The IDE main menubar, it's toolbar, and the script-editor menubar have been edited for brand-removal although there are a few additions as well. License checking and license upgrading related things have been removed from the Help Menu

The toolbar icons have been altered for better readability with dark UI themes and the toolbar texts font size can now be set (via the IDE Preferences) which may provide for better usability on smaller screens with limited screen realestate, or perhaps even to increase the size for larger, higher resolution screens. Palettes that are overlapping the Toolbar no longer inccorectly offsets the mouseLoc coordinates. The goal going forward will be to replace these pixel-based toolbar icons with resolution independent SVG graphics.

There is a new main menubar menu item to refresh the Toolbars rendering to force update any in-memory changes to it. There's also a new 'layout guides' that supports a feature that's been integrated (continue to 'new features' section for more on that). Some default acelerator key combinations for the IDE have been added or changed. For example command+shift+N (control+shift+N on WIndows and Linux) will create and new stack with the default height and width.

Tools palette

The tools palette previously used separate icon themes for classic controls on each platform that the IDE supports. These themes shared some common icons and the icons that were platform-specific were outdated showing appearances from old versions of an OS. For example the macOS theme had controls as they looked when run under macOS 10.4 (Tiger). OXT Tools palette has only a single icon theme for all platforms with more generic representations of the classic controls. More options have been added to the palettes settings allowing for wider column layouts, so you can now have a Tools palette that is 20 columns wide for example.

Eventually this tools palette will be replaced with an entirely new one with completely vectorized graphics (SVG) icons, and tabs that separate the various types of controls into sections (Widgets, Classic Controls, Graphics Tools) and then the tools palette will be made to be scalable to different sizes so that it will take much less screen real-estate on small screens, of be large enough for readability on high resolution (8K) screens.

Preferences

The Preferences stack has been added or modified to remove branding, remove the revOnline account related scripts, remove previous update mechanism, and a few new options were added such as a new IDE darkMode option. The stack graphics were also modified for an appropriate look when 'darkmode' is enabled.

Dictionary

Most references to 'LiveCode' in the syntax Dictionary (and there were many) needed to be edited or removed, which required that the Dictionary be rebuilt from the source mark-down (.lcdoc) files, along with some changes to the Dictionary stack's Javascript and xTalk scripts. Linux (x86-64bit) related script was added to enable inline-IDE Dictionary access. Previously opening the Dictionary on Linux 64 would launch the cached Dictionary HTML in the OS default web browser. New entries have been added that reflect the new status of this project independent of commercial interest. Entries have been added for other xTalk environments such as the open-source OpenXION JAVA based xTalk interpreter engine. Also, some APIs that were previously not included in the Dictionary, such as the Extension Utilities library and IDE add-ons API, are now included.

Resource Center

The Resource center was outdated by about 15 years. It was still branded as 'Runtime Revolution', the former name of LiveCode. Still this stack contains some still-relevant, useful reference information. In order to allow for editing the contents of this stack more easily, the stack was simplified with a dynamic navigation menu that is generated from the names of each card in the stack. Additionally a 'forwarder' stack was used to replace the old stack. A 'forwarder' stack is a stack that has the sole purpose of launching a different stack. This loads the new version resource center stack when the old original stack 'revResourceCenter.rev' is called for. With this method you can easily swap out the original stack for any other. This stack needs more work. The plan for this new resource center is to allow for it to be extensible by placing additional stacks in a particular directory. Hopefully this will allow for easy participation from community members in updating and expanding the examples and tips in this stack.

Extension Builder and Extension Manager

Besides de-branding changes there were some significant changes to Extension Builder and Extension Manager stacks and behavior scripts.

SVG Icons previews now display properly in the Extension Builder stack. macOS produced invisible meta files (filenames beginning with "._") no longer cause an erroneous 'no extension found' message when trying to compile an extension on other platforms.

Selecting a sample stack from the Extension lists menus now opens the stack as expected. Script extensions now display sample stacks in the same way that Builder extensions do. More list text options for the extensions lists have been added, allowing for better viewing on smaller screens.

The Snippets, and Sample Stacks tabs that were previously completely blank have been removed for now, but may return in some form once they're actually implemented.

The Extension 'Store' tab has been changed to 'Repo' tab and a new Extension 'store' has been created. This new OXT repository system is a Web stack (HTML5 deployed) and is based on simple tab-delimited lists and basic file formats (txt,png,etc.). This stack hosted via GitHub Pages and is 'Live' so this web stack can be changed at any time. For this to work more easily via GitHub, extension (.lce) files are base64 encoded and decoded for safe file transfers A new handler installExtensionB64Decode has been added to enable this feature. This feature is very much a work in progress. In the future a repo list will be added to allow for multiple user-repos by adding custom URLs or folder paths (for building up your own repo on a local disk). The goal will be to expand on this mechanism into a more holistic resource delivery tool, not just for Extensions but for Sample Stacks, Scripts, Graphics, Sounds, Documentation, URL links, including other open-source xTalk tools or any other sort of xTalk related resources.

There is also a new tab called 'Playground' which is an in-IDE web stack (Emscripten HTML5 Engine running inside a Browser Widget) for facilitating the testing of JavaScript and xTalk Scripts running within the IDE, A handler 'doXTScript' has been added that allows for calling out to the IDE Host engine to execute xTalk scripts in the context of the IDEs engine rather than inside the sandboxed web Emscripten engine. This mechanism allows for two-way communication between a web stack and the IDE. This is the stack I've personally been using to test the capabilities of the Emscripten xTalk engine as it was left to the community. I've also used this stack as something of a prototype for wrapping the Emscripten engine with the Electron JavaScript-based desktop application engine, which has potential to allow for deploying stacks as standalone applications on platforms such as FreeBSD that may not be directly supported by the OXT IDE standalone builder. I believe there is enormous potential in the combination of the Emscripten xTalk Engine and the JavaScript based Electron Engine. For example multiple instances of the Emscripten Engine could run in there own threads using JavaScript to communicate with the other instances, enabling platform agnostic multithreaded multiprocessing for your open-source xTalk projects. This web stack is also hosted on GitHub Pages 'live' so its content may change at any time.

HTML5 (Emscripten) Engine

Some of the HTML5 (Emscripten) deployment output has been modified to remove user-facing branding. In the future there will be more options in the HTML5 deployment tab of the Standalone Builder, with options options such as including your own icons or include some additional javaScript libraries in the HTML5 page output, and perhaps even options for wrapping the output in an Electron.JS wrapper application, perhaps with available options for any platform that the Electron engine can run on (such as FreeBSD). I see enormous potential for the community in this idea as the Emscripten xTalk Engine as perhaps the most platform agnostic engine available (Java based OpenXION engine is also rather platform agnostic). Some future goals for the Emscripten engine going forward will be to compile more of it to platform independent web-assembly byte-code to try to increase speed, and implement more HTML5 APIs, facilitate even easier JavaScript <-> xTalk interpreter crosstalk. Ultimately I want to be able to wrap the standalone HTML5 output with a JavaScript desktop app wrapper such as Electron or Deskgap. Hypothetically, running xTalk inside a JavaScript based app wrapper such as Electron in this way offers worlds of new possibilities, such as 'native' app deployment on platforms not directly supported by the IDE, or having multiple instances of the engine running in separate web view processes, calling back and forth to each other via a javascript bridge for applications that require more threads for multiprocessing, as well as gaining first-class access to all of the Node.js JavaScript ecosystem.

New Features

macOS Native Tools library

macOS Native Tools is a library of handlers that wraps some of Apples AppKit framework and other APIs that provide additional 'native' xTalk scripting capabilities for macOS. The library was added to the OXT IDE in order to provide macOS native 'darkMode' to the entire IDE (can be applied on each individual window level as well). This library also provides a bunch of other handlers you may find useful on macOS, such as the ability to set an app's Dock Badge Text (often used by apps to show a count of something, such as number of new email messages), set a files mac Type and Creator Codes (for the retro users), get an accurate read of the macOS version (working around incorrect reporting from the OS itself, returns 11.x for Big Sur instead of 10.16.x for example), and more. See the sample stack included with the library for demonstrations. You can launch sample stacks from the Extension Manager (bugs fixed), and more! More macOS native capabilities will be added in the future.

Layout Guides

Graphics layouts 'smart' guides (as implemented by several community members) has been integrated into the IDE. These guides are very similar to smart guides found in vector drawing application such as Adobe Illustrator. The are very helpful for manually aligning objects in a graphical interface layout. You can enable and disable these Layout Guides from a new toggle menu item in the view menu of the main menubar.

General Music Library and related Builder Extensions

In homage to HyperCard, and reflective of my personal interests, xTalk 'playSentence' musical syntax has been incorporated into the OXT IDE. One of the first things that I looked for when I discovered LiveCode sometime around 2014, shortly after the open-source campaign, was presents of xTalk 'playSentence' form of the 'play' command. As teen in the mid-1980s, and coming from previously having only played around with AtariBASIC a bit, I thought it was just insanely great to have such a simple text-based musical notation and sound sample playback engine in a user-centric programming environment, I was instantly excited. So it was a big disappointment for me to find that LiveCode had only most basic built-in sound-sample playback support with no support for playing samples at different pitches or note durations. The rest of the IDE was pretty much what I'd expected in an xTalk that has continued to evolve, and had some modern syntax additions one might expect such as support for arrays and full unicode text support (a major goal of the open-source campaign), and yet it didn't stray too-far from its traditional (HyperCard clone) xTalk roots which I thought was great! And so I set out to enable a 'playSentence' replacement (and more, I really wanted a modern replacement for ancient HyperMIDI Xternals or similar).

It turned out there was already a well written, public-domain, open-source xTalk library from 'UDI' called PlayPMD/makeSMF that had implemented 'playSentence' compatible syntax with some MIDI related additions. This library was written more than 20 years ago and was available for HyperCard, SuperCard and Runtime Revolution (MetaCard), and used Quicktime Musical Instruments as playback engine, but it still works great, better then ever in fact. The only problem was with relying on Quicktime for playback as Apple eventually dropped support for Quicktime Musical Instruments and later dropped support for QuickTime altogether. I tried several alternate playback methods such as using FluidSynth command line interface (via xTalk 'Open Process' syntax), then later implemented playback via Web MIDI APIs in the BrowserWidget (which required installation of a Web Browser-Plugin), until finally Extension Builder and its Foreign Function Interface were introduced. I spent the next few years learning the Builder language and developing the Extensions that I needed to facilitate direct playback from APIs added into the IDE.

Several playback engine extensions were created. The first was an extension that uses Apples simple AVMIDIPlayer API (available on macOS and iOS), the next one used Apple's CoreMIDI API directly (send MIDI notes to GarageBand from your xTalk scripts). Next I wrapped Apples AVAudioUnit Sampler (realtime software instruments), which can load Logic/GarageBand EX24 instruments as well as SoundFonts, DLS banks, and plain old wave or aiff sound files. Next I worked on an extension that (attempts to) load ANY Software Instrument AudioUnit Plug-in, including a plugins embedded UI (which is currently rather crash prone, needs work).

Finally an extension wrapper for the open source cross-platform software sampler instrument library and MIDI playback engine called FluidSynth was created with the help of a few community members, which enabled pitched sample playback on all IDE platforms (should also work Android deviced but that is untested). The library is not a complete wrapper of libFluidSynth, for example FluidSynth now has a sequencing API that has not yet been wrapped.

To tie together all the playback possibilities a new Script library extension was library called 'General Music' was created. This library contains UDIs xTalk 'playSentence' notation interpreter (+ updated) and outputs to the default output API based on the platform the stack is being run on. There is also a bunch of MIDI and SoundFont related scripts handlers for things like reading sound patch names from SoundFont or DLS soundbank files, converting MIDI file data into a human readable format, handlers for converting between MIDI numbers and to and from human readabile formats, etc. This library will scan for soundBanks files in the appropriate directories for the current platform and load a default sound-bank for the current OS (for example the QT Musical Instruments GM bank, which still exists in macOS, now part of the CoreAudio.framework) or the included fallback soundfont (FreeFont.sf2).

As another homage to HyperCard, the General Music library also includes a HyperSounds.sf2 bank which contains the three sample sounds from HyperCard; Boing, Harpsicord, and Flute, modified from the SoundFont conversion from Becky Bettencourt (creator of OpenXION). So in OpenXTalk IDE from the message box you can enter 'PlayPMD Boing, C4q' and hear that old familiar HC sound played back at middle-C (octave 4), for a quarter note length, at a default tempo (120 beats per minute).

I plan to continue adding all sorts of music and sound related handlers to this library. In addition I'll soon be adding music related graphics, SVG Icons (corresponding to playSentence durations: w,h,q,e,s,t,x), SVG Icons for play, stop, pause, fast-forward, etc.

Additionally, there is an on-screen clickable Piano widget control included with the OXT IDE. This widget was designed for use with these playback APIs. 'mouseDown' in this widget also sends a 'noteOn' message and 'mouseUp' sends a 'noteOff' messages to the widget object with parameters containing the clicked noteNumbers (60) and a second parameter with noteNames ('C4') which can then easily be trigger a playback engines *NoteOn and *NoteOff handlers and displayed as a name (for humans). This widget has lots of features for styling and colorizing the look and individual keys on the piano, playing 'chordFormula' chords, locking input to a musical scale, and more!

Notes from the author:

OpenXTalk DPE IDE vs. OpenXTalk.org

There is a difference between OpenXTalk DPE (the IDE) and OpenXTalk.org as an idea. I intend to keep working on the LC community edition based IDE, with the help of others or not, but I also think that OpenXTalk.org should be about promoting ANY and ALL efforts to keep xTalk alive as a language that is available to the general public, regardless of economic circumstances and so with a focus on open-source xTalk(s) in particular. I've written some xTalk (+AppleScript) scripts recently targeting the excellent OpenXION xTalk interpreter (runs on Java VMs) by Rebecca G. Bettencourt, and later did an experiment to compile the OpenXION 'engine' into a 'native' executable (native in that there's no longer a need for a Java VM to be pre-installed on the system). As another example, I also find ///_HyperScript to be very interesting conceptually, ///_HyperScript is a library that enables xTalk syntax mixed with web (DOM) elements. There are multiple other C++, Java, or Web/JavaScript based HyperCard clones that I've been looking into (StackSmith for example) as well as a small community of retro-user still playing around with HyperCard things purely for retro-fun. I would welcome displaced SuperCard, Toolbook, Lingo, etc. xTalk users as well. I want OpenXtalk.org to be open to, and supportive of, NOT ONLY the legacy LC Community Edition, but all of these things as well!

Criticism

When you criticize this work (and it already has had criticisms all along the way), please keep in mind that I do not even consider myself to be a professional programmer. I am a graphic artist (I'll accept 'graphics technician') by trade, who just happens to have had a lifelong interest in programming, xTalk and related scripting in particular, who has picked up on some lower-level (C, C++, Objective C, etc.) programming along the way. I'm working on this because nobody else would, I think I have a diverse enough skill-set to do some good here, and I believe there's untapped potential for xTalk and the democratization of application programming. I also believe there can be an xTalk community built around any/all xTalk implementations (but open-source ones in particular), available to anyone in the general public who is interested. I'm donating time away from other personal interests and sometimes even from my family in order to work on this for no financial gain, purely altruistic reasons. That said I'm all for taking in any valid criticisms as long as they're constructive criticisms.

Thanks

I'd personally like to thank all of the community members that have been encouraging, added fixes or scripts and stack content, contributed to the conversation, offered prodding (Richmond), etc. along the way. In particular I'd like to thank Seth Morrow for not only providing a spark, but for hosting our forums.

Help Wanted

Lastly, PLEASE HELP! OpenXTalk needs all the help it can get! You DO NOT Need to be skilled C++ developer (although that sure would be helpful) in order to contribute to OpenXTalk IDE! The IDE is written in xTalk script so experienced scripters can help with that a lot! In fact NONE of the changes or additions I've listed above required rebuilding the xTalk engine(s) from its C++ source code (which is one of the next goals for OXT IDE, and I do believe I have everything that I need to do that now). The Dictionary is built with JavaSCript (mostly) and parsed from a fairly simple markdown format (.lcdoc). You could also contribute stacks, libraries, or widgets you've written that you think may benefit the xTalk community as a whole. You could even just join the conversations in the forums (forums.openxtalk.org) and provide your perspective or links to articles about xTalk or natural-language programming or related.

Paul McClernan

October 2022