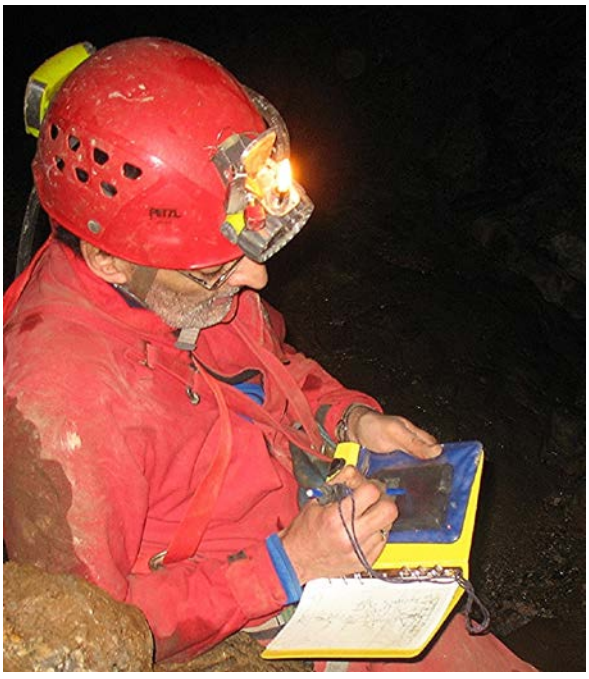
Auriga – Looking Back at 15 Years of Development

Auriga was the first caving software specifically designed for underground use. Its adoption by many expeditions has improved the quality of their survey data by favouring in-cave corrections of survey errors, while also helping orient explorations. ***Luc Le Blanc****, software designer, reflects on his 15 years of developing the Auriga freeware.*

In 2002, our "Mexpé" [1] expedition was seeking a low-power survey data processing solution to be used at its base camp in the Mexican jungle. Our options back then were a programmable pocket calculator and its tedious interface or a laptop for which we had no solar panel, only a car-rechargeable lead-gel battery. But some of us owned a Palm OS personal digital assistant (PDA) powered with a pair of AA alkaline batteries, which could run a spreadsheet program. So, I searched for a ready-made caving spreadsheet, and instead found a small (40Kb) piece of software written to test a prototype azimuth and slope sensor box over a (wired) serial link.

In 2017, Auriga [2] is now full-fledged cave survey software still running on Palm OS handhelds, but also Android devices under the StyleTap [3] emulator. The software handles numeric survey data, manually-input or received over a Bluetooth link, provides a graphic rendition of the cave map, allows on-screen sketching of cave walls around the line plot, imports/exports from/to various cave survey or drawing software, manages cave networks, features loop closure and GPS tracking over the underlying cave passages and supports numerous measurement units, options and user preferences. This is way beyond the initial requirement! What happened between those dates?



Daniel Caron using Auriga on expedition

From Debug Tool to Main Program

While mostly designed to help test and debug his prototype sensor box, Martin Melzer's Auriga program had the ability to display a simple cave map [4]. Features were limited, but this was already lots more than we had hoped for. The program demonstrated that under a very lean operating system, trigonometry and graphics could be done at usable speed on a 16MHz processor; this gave me the impulse to shift our goal of a base camp survey data processor towards an in-cave solution to input, view and fix survey data on a handheld device. As a software designer by trade, I thus offered help to improve the software with regards to its user interface and list of missing features. I soon learned Martin had given up on the sensor box project, and the source code was all mine to work on. I bought the Palm OS Bible, found a copy of the CodeWarrior Development Environment, and I was in business.

Palm OS is an easy-to-learn small event-driven operating system [5]. Its natural language is C, most likely because the constant memory allocations of an object-oriented language like C++ would be too much of a burden for small devices.

As with most human endeavours, I had no prior idea of the magnitude of the effort I would end up putting on this project!

Mexpé as a Test Bed

From the start, the Mexpé project seemed an excellent test bed: we had a few multi-lingual teams (French-speaking Quebecers, British and American guests and local Mexicans) surveying numerous caves, sometimes over several years, some eventually connecting. Each team would have its own habits and gear, and would sometimes switch from metres to feet. If Auriga could handle all this, it should satisfy everyone's needs, provided the software adapted to users, not the other way around.

The problem of mixing measurement units was solved through the use of survey sessions: each survey shot between two survey stations belongs to a session describing the tools used, their survey units and the contemporary magnetic declination. This way, survey data could be stored in a unit-less fashion, as read on the instruments, with conversions or calibrations applied only when computing Cartesian coordinates from polar data (i.e. survey shots).

In 2003, I was back at Mexpé, this time with an improved Auriga. But one problem that immediately arose was the inability to compute caves surveyed in a discontinuous fashion: a team could start exploring and surveying from the cave entrance, then stop surveying and push to the bottom of the cave, and return another day to survey from the bottom back to the end of the original survey. This clashed against the Toporobot-inspired constraints originally implemented in Auriga (Toporobot, an early cave survey software for Mac OS, imposed surveying with numerics-only station names in a strict gap-free sequential order).

After days of ruminations, I solved the problem by implementing an algorithm that traverses the cave graph in alternating direction, each time computing still uncomputed stations linked to now-computed ones. The process stops when every station is computed, or nothing changes anymore, allowing as a bonus to detect orphan survey shots (not linked to the 'rest' of the cave) or

loops (still uncomputed survey shots linking already computed stations).

Going Public

While Auriga was initially meant for Mexpé, it made sense to broaden its use given the amount of time already invested; I viewed it as returning something to the community. Thus, in December 2003, the first public beta of Auriga was made freely available to everyone in three languages. For the first time, a cave survey software was designed from scratch to be used underground, replacing paper for the input of numeric data. Data was input one survey shot at a time in a self-validating window, with the help of a custom keypad that could be used with gloves. Computations worked both in batch or incremental modes (as new survey shots were added), the expected values for backsights were displayed in real time, an Assistant offered showing the updated line plot after every new shot, and surveyors could readily match their work against reality, and fix survey errors while still in the cave. Designed for a small screen, the program offered features like a clipping mask for uncluttering the screen. A growing number of enthusiastic early adopters helped improve the software with bug reports, suggestions and sample data. 35 more versions were then released over 2.5 years (more than once monthly) until version 1.00 on July 23, 2006, deemed stable enough to be used reliably by all. A few people have asked me whether the source code would be published, but I feel this question is mostly rhetorical, as nobody ever offered to help with programming. And given the amount of time I devote to this task, I'm not sure any helper could follow the expected pace.

To many users and observers, the ultimate feature would be to support on-screen sketching of cave walls and features around survey shots. I pondered this idea for years, knowing it would be hard and complicated, as I had no access to any computer graphics library whatsoever. As a matter of fact, aside from the basic OS APIs to draw windows and UI elements, open files and manage low-level data communications, the only external library Auriga uses is a set of math and trigonometric functions (only square root, sine, cosine and tan are used). So, I postponed this ordeal and concentrated on other features that would prove convenient underground.

The rationale behind Auriga was that it should be as flexible and generic as possible and support any feature deemed useful underground even if these features were not implemented in desktop software: it was left up to each user to decide which feature he would use, or dismiss, depending on his target desktop software. Or he could ask the target's developer to implement these features. One example is splay measures, used to complement the traditional passage widths and heights in more complex situations, or to assist with sketching. Sadly, several cave software programs do not support such supplementary measures.

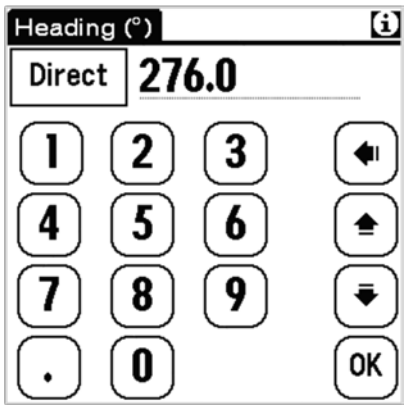
Auriga running on the emulator

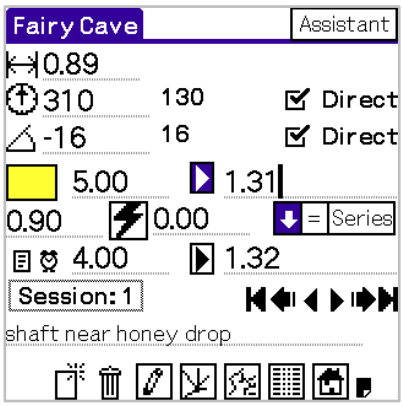
Given the time and effort it takes to survey caves, data security was a concern, and early on, Auriga implemented the ability to 'beam' its caves over the built-in infrared link, and later to back them up on the removable memory card. These features were later expanded to support Bluetooth and the merging of cave data when two survey teams meet each other underground.

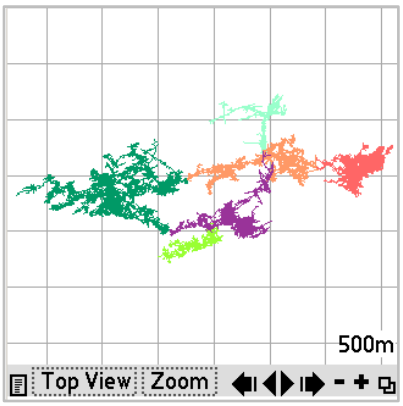
Loop handling was improved beyond their basic detection, with the ability to zoom on a loop in the map and analyse it to try finding a single possible blunder explaining a larger-than-expected error. The concept of loops was also extended to passages joining GPS-positioned stations, such as an entrance and a radiolocation site. The path-finding algorithm implemented for loops was then re-used to show the shortest path between any two stations, a feature I would much appreciate when trying to find a shortcut back to the start in Lechuguilla's maze!

Auriga being a field tool, it made sense a GPS could be connected, both to input the position of entrances and surface features as well as to follow on the surface the underlying passages in search of new entrances. So, I delved into the mysteries of geodesy to support latitude/longitude as well as UTM (Universal Transverse Mercator metric grid) and Lambert coordinates across a wide range of geodetic datum. The ability to geo-reference caves made cave networks easy to handle and render graphically as a single map. Conversely, once underground, it became simple to view whether a passage under exploration led towards another known cave or a surface feature like a fissure or a sinkhole. Of course, this georeferenced line plot could also be exploited outside the PDA thanks to GPX and KML (Google Earth) exports.

During those years, the Palm ecosystem was still thriving and Auriga had to keep pace with new hardware features like faster processors (400MHz!), high-resolution colour displays, a virtual input area and Bluetooth connectivity. In 2004, taking advantage of a free plane ticket my friend Chris Chénier couldn't use, I even attended the last PalmSource Developers' Conference, a $1000 folly (conference and hotel) I made for the benefit of my users, gaining more insight on arcane aspects of Palm OS coding and debugging. Perhaps unsurprisingly, of the 900





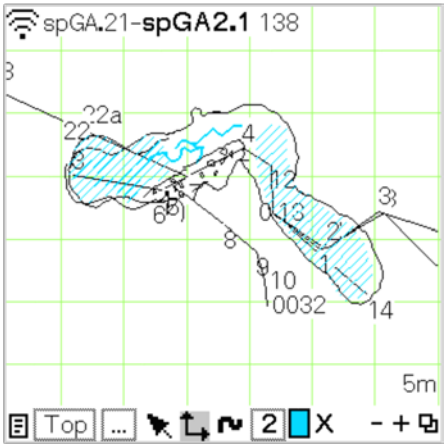


Three screen captures showing on-screen keypad, shot data entry screen and a map of Lechuguilla's 42,323 shots, assembled from 7 subcaves imported from Compass

developers taking part in this three-day event, I was the only one writing freeware, an "anomaly" as someone told me.

Sketching goes Digital

In 2011, I felt ready to tackle on-screen sketching. But contrary to suggestions I received that I could first implement a bitmap (raster) solution and make it vector-based later, I decided it had to be vector-based right from the start. Indeed, a bitmap approach was incompatible with my goal of a sketch that was editable (i.e. part of a curve could be modified, cut or extended instead of requiring to retrace it entirely) and morphable (i.e. that could self-adapt to length or angle changes made to survey shots after fixing an error or applying loop closure). So, I went to McGill University, my alma mater, to peruse technical literature on computer graphics. Alas, the basics are now hard to come by as



Sketching

most low-level procedures like converting an array of pen coordinates into a spline [6] are now implemented in widely-used code libraries. Unfortunately, those libraries not being available for Palm OS, I had to code from scratch. Furthermore, after much reading, I realized that splines would make it pretty complicated to implement curve editing, like when correcting part of a curve drawn earlier. This simple action we commonly perform with a pencil and eraser requires determining the contact point(s) between the stylus and a spline equation. Without the expertise and personnel of an Adobe Illustrator development team, I settled for polylines, i.e. a set of straight lines emulating curves. Fast algorithms for converting pen coordinates into polylines were easily found online, and determining the intersection point between a polyline and the stylus was within my mathematical abilities. I could thus implement curve edition, splitting or extension, and conversion into closed polygons. All this made its way into version 2.00 published in July 2011. Further releases added line styles and polygon fills to speed up the sketching of pit lines and other structures. The sketch was soon added to the existing DXF and SVG exports.

Getting the Data Out (and In)

The standard procedure to exchange data with a desktop computer under Palm OS is called a *conduit*, a custom subprogram launched by the HotSync Manager when the PDA initiates a data link with the PC. Almost right from the start, Chris had written a conduit to exchange data between Auriga and Compass and Visual Topo. These targets were chosen because they were respectively North America's and Europe's most popular cave survey software. Toporobot came on later, because it was the only native Mac software at the time. Chris had imagined synthesizing format exchange rules as an XML grammar to speed up conduit writing, but this proved more complex than he thought. As time was passing by, and PCs moving to a 64-bit architecture, running the HotSync Manager became more difficult, so I decided to write import/export procedures within Auriga, using the memory card as the exchange media. In a matter of a few months, I re-implemented the already supported formats, and added Survex, Therion, Walls and a generic CSV format.

The cave data in Auriga is stored as a 'Palm OS database', actually a set of records (one per survey shot) sorted by their station names. This format consists of publicly documented C structures. Of course, an embedded version number allows detecting incompatibilities with more advanced versions, triggering an automatic data migration. For efficiency reasons, the data is binary and often stored in compressed form, such as hundredths of length units (i.e. centimetres or 0.01 foot) in 16-bit integers. One must not forget the earlier target devices had only 2Mb of RAM to hold both the program and its caves. But despite the larger memory size of newer devices, Auriga has always remained frugal in its use of resources, and code is re-used or made re-usable whenever possible. Despite 44 windows and dialogs, 205 alerts, extensive computation and rich UI features, the current executable only weighs 1.2Mb, about half the size of the average flashlight app for Android devices...

The recent Auriga versions mostly polish existing features or fix unfortunate bugs. Its e-mail list joins nearly 200 subscribers worldwide, most of them feeding a few fellow cavers. The software is used to map small caves, archaeological sites, urban underground rivers, multi-sump caves as well as extensive high-altitude cave networks. Spain seems to be the epicentre (hence the recent addition of a Catalan version), with several invitations to teach classes, and now numerous caving clubs setting up classes to spread its knowledge.

The software is now reaching a plateau, as there is only so much worth implementing in software designed to be used on a handheld device underground. As a matter of fact, my most common answer to users' requests is "It's already there!". Auriga is now so feature-rich that some users report going directly from Auriga to a sketching program via the SVG or DXF exports, skipping desktop caving software altogether, a situation I had not envisioned initially, the design goal being to complement desktop software, not replace it. This feat is quite amazing given that I get to survey caves about twice a year during week-long trips to Mexico, Lechuguilla or Crete. With its limited limestone beds laid out in valley floors, a modest vertical drop and a relatively recent deglaciation, Québec has few caves over a km long. Some remote areas offer a potential for major discoveries, but weekend trips are instead devoted to digging. Auriga is thus a thought experiment for which Chris and I reflected enormously about how the tool should behave and which options to support to offer as much flexibility as possible, then pleasing most everyone. The only drawback to this flexibility is a large number of options, which may discourage some. Hence a 'safe' choice of default options favours an easy quick start with the program.

The next major move would be to port Auriga to Android so as to run natively on smartphones and tablets, thus enjoying the full hardware capabilities of these devices. But this is a major task that I must consider wisely before undertaking, as Android development is way more complex than under Palm OS, and I already devoted a lot of time to this freeware. In the meantime, Palm OS devices in mint or excellent condition can still be found on eBay and similar websites, for less than the $50 price of the Palm OS StyleTap emulator for Android, and without risking a fancier device in a hostile environment.

References

[1] Mexpé is a recurring caving expedition in Mexico's Sierra Negra held under the auspices of the Société québécoise de spéléologie (Québec, Canada)

[2] **speleo.qc.ca/Auriga**

[3] **styletap.com**

[4] Melzer, Martin (2003), *How to Design an Electronic Surveying Instrument*, CREGJ **54**, pp 12-15

[5] Melzer, Martin (2002) *Palm Programming for Embedded Applications*, CREGJ **48**, pp 28-29

[6] Splines are sets of polynomial curve equations that each apply on a given range of coordinates and whose segments meet with a high degree of smoothness, thus producing smooth-looking lines with varying curvature

CREG