UltraDynamo Case Study

By William Van Winkle

From *Top Gear* to Top Winner

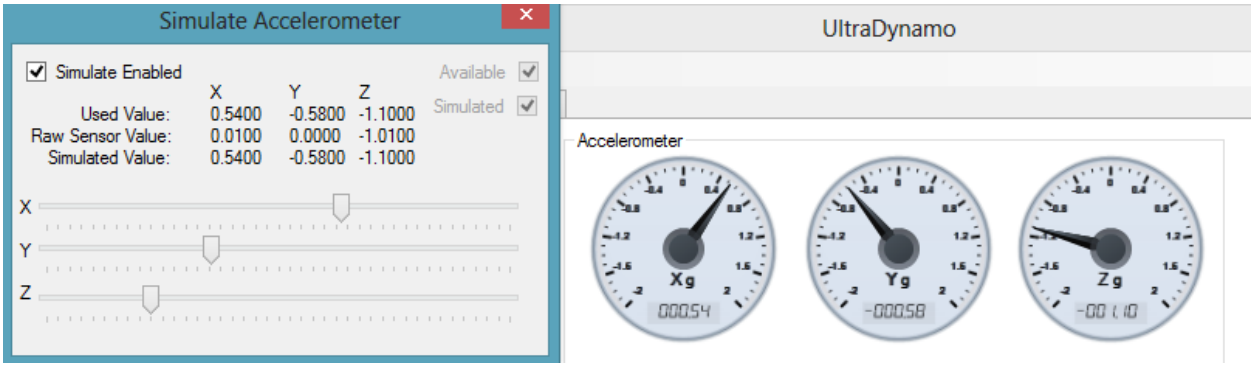
By day, David Auld is an Offshore Installation Manager (OIM) in the oil and gas industry. But when the production platform is humming along without him, Auld indulges his hobby as a devout "petrol-head" (car enthusiast). He also finds time to feed a passion for programming, which led to him earning a 2012 BSC Honours Degree in Computing. Surprisingly, these three facets of the native Scotsman all converged when Auld won the Entertainment Category of the Intel® App Innovation Contest.

Source: http://www.mbusa.com/vcm/MB/DigitalAssets/Vehicles/ClassLanding/2013/C/Coupe/Gallery/2013-C-Class-Coupe-Gallery-009_wr.jpg
With UltraDynamo, art may have copied life when developer Dave Auld took inspiration from his own Mercedes console.

Auld has been a CodeProject member for nearly 10 years. He also takes pride in owning a Mercedes-Benz C63 AMG sedan, the latest in his long line of personal sports cars and one particularly blessed with a graceful and classic dash console. Perhaps this was in the back of Auld's mind when he noticed CodeProject advertising the Intel App Innovation Contest. He read through several of the proposals and thought, "There must be *something* I can come up with..." From there, all it took was watching a *Top Gear* rerun featuring the Bugatti Veyron and its horsepower indicator. "How did Bugatti do that?" he wondered. In working toward an answer, Auld stumbled further into a series of questions and revelations that led to his award-winning success only weeks later.

The UltraDynamo App: Form and Function

UltraDynamo is a Microsoft Windows* Desktop application that uses many of the Ultrabook™ device platform's sensors to provide motor sports enthusiasts with performance data about their vehicles. As shown in the screen capture below, UltraDynamo offers a range of readouts, including x-, y-, and z-axis accelerometers, a compass rose, a speedometer, inclinometers, and gyrometers. These might be presented as charts, pictures, numeric readouts, and so on. The data for each of these springs from various Ultrabook device sensors, including the accelerometer, gyrometer, inclinometer, and a Global Positioning System (GPS) sensor. In short, UltraDynamo presents a configurable on-screen dashboard.



Without real sensor data on hand, Auld saved ample development time by simulating input values. This screen capture shows a typical simulation dialog box and its effect on the main dashboard.

In looking at the application, Auld's priority was clear: Keep the front end as clean and simple as possible to minimize key entry by the user. (Obviously, requiring manual interaction while the user is behind the wheel would be undesirable.) In the same vein, he understood that different users would come to the app with different needs and priorities. The UI should reflect that. Thus he broke out individual readout functions into separate window elements that users could reposition and resize as desired.

For Auld, this UI simplicity should also be reflected in the program's responsiveness. "Usability is key," he said. "Users want that reward: when they click, the app does what's expected. That's what will keep them wanting to use the program."

The UltraDynamo app relies on a flexible dashboard interface featuring a range of gauges, including compass heading, acceleration, speed, and horsepower.

Auld developed UltraDynamo on a pair of PCs running Windows 8 Pro, one desktop and one laptop. Neither had any sensors but both had copies of Microsoft Visual Studio* 2012 Pro. Once his concept application was accepted, CodeProject contacted Auld to confirm that he felt he could provide a working application for the competition. When both agreed that it was feasible, CodeProject sent Auld a sensor-equipped Ultrabook device with Windows 8 Pro and Visual Studio 2012 Pro. Auld noted that in order to

keep all of his development systems “in check,” he used VisualVSN Server* as a source code library. This library is hosted by a cloud provider on a Windows 2008 R2 virtual machine.

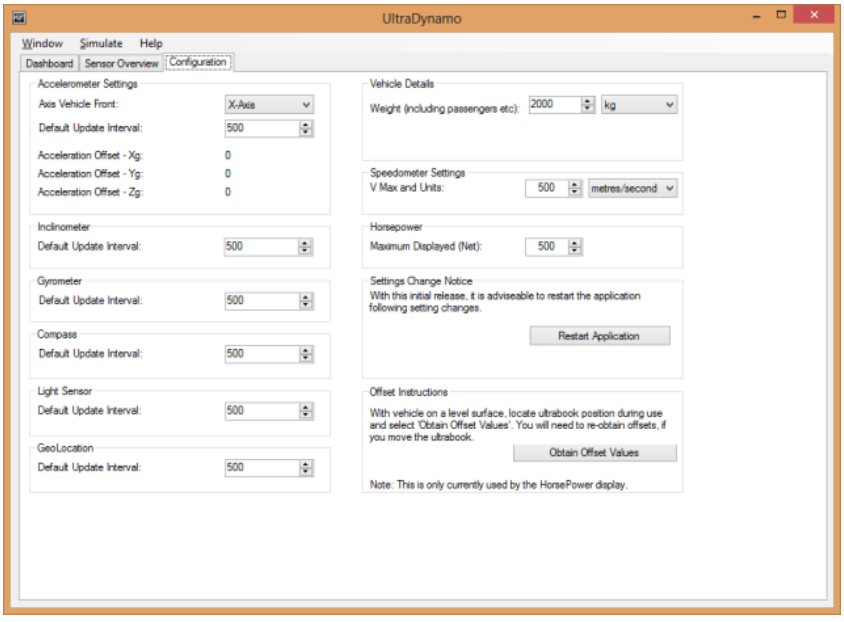
“I used ankhSVN plugin for Visual Studio,” he added. “It was a simple case of checking in any code changes on one system following any edits, then updating the source to the latest version on the others. This worked well as a way to manage the source from a multi-system single developer point of view.”

Challenges Addressed During Development

One of the first obstacles Auld had to conquer was a lack of resources offering suggestions on how to handle sensor data. For example, after getting a temperature value, what does the programmer do with it? Ultrabook devices and their many sensors are relatively new to the market, so there isn’t a large bed of third-party examples and advice to follow beyond Intel’s own Ultrabook™ and Tablet Windows* 8 Sensor Development Guide and the Windows 8 code samples Intel offers. Auld had to figure out many of the answers on his own.

His first such problem was the original program interface. It was, as he put it, “just a bunch of random numbers on the screen.” He needed gauge controls to mimic actual dashboard readouts. At first, he tried to design these on his own, but it soon became clear that there wasn’t enough time to build what he wanted from scratch. He searched the Web, cast about the CodeProject site, and finally unearthed a license-free dial control called Aqua Gauge, written by Ambalavanar Thirugnanam. This dropped easily into Auld’s code and became the backbone on which the other UltraDynamo controls were built.

Auld also found that frequent accelerometer sensor updates were causing an event flood, which in turn stalled the interface graphics. Through trial and error, he worked to change the time intervals for eventing data. Finally, he got the display working and stable, although he hopes to return to it for further tweaking. Rather than poll data on a fixed interval, Auld wants to see the app work on a more intelligent feedback loop wherein the app doesn’t request more data until the graphics system is ready to handle it.



UltraDynamo’s Configuration tab offers a range of input frequency settings for the Ultrabook™ device’s various sensors.

As mentioned earlier, Auld's day job experience played into his UltraDynamo development. After having his proposal accepted for the Intel contest, Auld had to wait to receive his Ultrabook device, and during this time his job required him to go offshore for many days. Fortunately, his background as a control systems manager found him frequently building simulations so that the graphics could be tested without requiring the production plant's systems to be available. The same methods applied here. He wrote the graphics first, created a dummy set of data, and worried about the sensors later.

"It was simple," said Auld. "Put a bunch of sliders onto a form and group them into the relative component, whether it was accelerometers, gyrometers, or whatever. That allowed me to manipulate the graphic as part of my testing without actually having hardware sensors available to me. That was a significant benefit. Otherwise, I would have had to spend several days writing code, then get the Ultrabook from Intel and find that nothing worked. I would have lost a huge amount of time. Let's program for the graphics and write it in such a way that I can just plug in the sensors at a later date and, in theory, it should all work nicely."

Auld had to take some educated guesses on the data boundaries the Ultrabook device sensors would generate, but once he finally received the device and got started, it all worked fine. Fortunately, his long career and ample experience with touch and sensor development helped him to steer clear of any major issues in these areas.

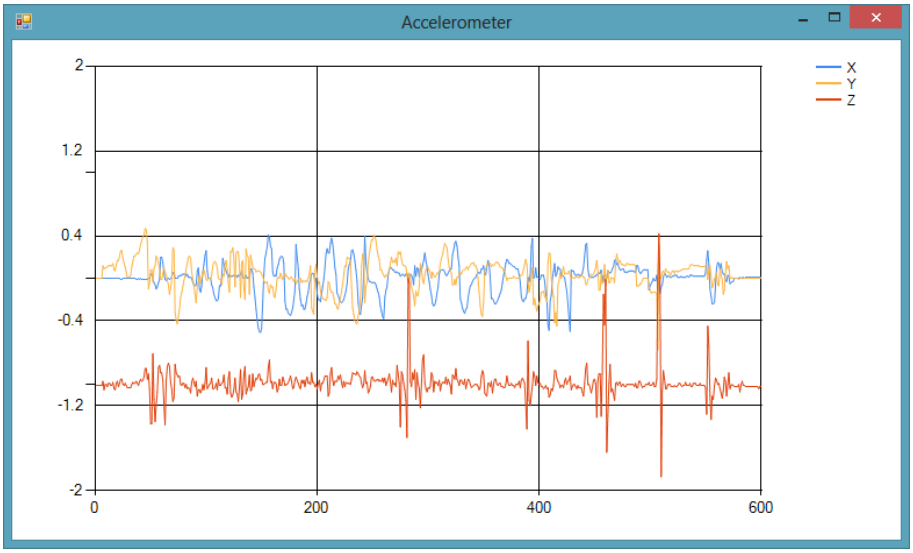
UltraDynamo's last major hitch revolved around the MSI installer required for submission to the Intel AppUp® center. Originally, Auld intended to generate the package from the InstallShield* Lite tool that comes bundled with Microsoft Visual Studio 2012. However, no amount of banging his head against the application helped him understand how to generate an MSI package directly. No matter what he tried, all he could get from the program was an .EXE installer, which the Intel AppUp® center wouldn't accept. Finally, Auld did find a way to "double-install" into an MSI package, but the Intel AppUp center wouldn't accept that either. Apparently, examination by Intel techs in a test environment revealed that "the shortcuts that the application installed weren't announced shortcuts."

"To this day, I haven't got a Scooby what that means," admitted Auld.

Fortunately, Intel came to Auld's rescue. Tech support staff sent him an alpha version of a tool they used internally for app store packaging that relied on WIX* as its underlying toolset for generating installer packages.

"After working out how the Intel-provided app ran a couple of the WIX underlying commands to generate the MSI package, I took the XML file that the tool had created and used it as a foundation. I tweaked the internal XML nodes, got my shortcuts displayed on the screen, and then manually ran the WIX underlying commands to generate the MSI package. This then went through verification at Intel without any issue."

All told, Auld spent about four weeks designing UltraDynamo while working a full-time job. This was broken up with all-consuming work on his production platform, waiting for verification from Intel on different code fixes and so forth. It was a tense, utterly time-constrained process, but it forced him to focus on what was essential for meeting milestone deadlines and to find solutions within his limitations. The lessons here for a part-time, lone programmer were significant.



Simple but effective, this graph shows UltraDynamo's graphing of real-time data from X, Y, and Z axis accelerometer sensor inputs.

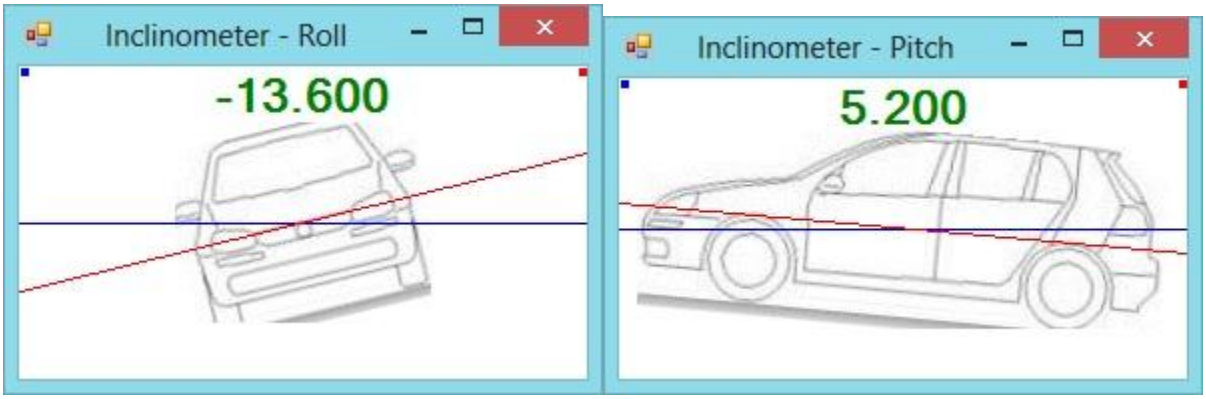
Lessons Learned, Advice Given

UltraDynamo went on to win the Intel App Innovation Contest's Entertainment category, but that doesn't mean the application is finished. Auld said he had to leave many ideas on the drawing board because of time constraints, and the UI that did emerge was largely tailored to his own interests. He would like to see the app develop "workspaces" in which users could customize their dashboards and save them like profiles. He would also like to find more professional-looking gauges before commercializing the software.

UltraDynamo's development was much like any other app development, fraught with its own complications, delays, and breakthroughs. "Maybe it's frustrating," said Auld, "but it does help you to think for yourself, and to try things and dig deeper. In the process, you become proficient."

He encourages other developers to be willing to learn, experiment, and fail. As an apprentice, when learning the systems on a new platform, Auld had to figure out all of the plumbing and parts and systems on his own. Supervisors would steer and make sure he "didn't do anything stupid," but it was an environment for the inquisitive, adventurous, and self-motivated.

Auld says that such a mindset is becoming increasingly rare in a time when young programmers would rather be spoon-fed code than take five minutes to write something and see if it works. On CodeProject, Auld tries to point people in a direction and encourage them to reverse-engineer what other people have done instead of saying, "There's your 25 lines of code. Get on with it." Every project involves a learning and research phase. Expect it, don't look for shortcuts, and keep the results of these learning processes in a personal code library.



Roll with what you've got. Given his time constraints, Auld had to use some generic car images as part of the interface's readouts. He hopes to expand this image set in the future.

Even before starting to code, Auld recommends that developers write out the app they have in mind as a narrative. Approach it as a technical article. By creating at least a plan in bullet-point form, it forces the developer to break down the application's structure and functionality, which in turn helps offer more guidance in the application's development. Writing an application as an article will force the developer to think about what he or she is trying to convey to the end user and the ways in which those priorities can be best communicated.

Finally, Auld encourages developers to "just dive in." Try and fail. Don't be afraid to ask questions and get involved on sites such as CodeProject. Auld admits to being little more than a silent trawler for his first seven or eight years on the site. Armed with enough years of slow but sure learning, he was finally ready to become more active and give back into the community. He adds, "That is an important thing for people to do. Don't just take all the time, but give back, as well."

Resources

Auld provides extensive details on the processes and tools he used in constructing UltraDynamo in his five-part CodeProject article. This shows his path from Visual Studio setup through code signing and packaging. Along the way, he also investigated online coding tool reseller ComponentSource and, as noted earlier, resources found at CodeProject ultimately formed the foundation for UltraDynamo's interface.

Auld stresses that he couldn't have won his contest category without help from Intel's forums, Intel tech support, and, most of all, the developer community. "Without the inspiration and help of notable gurus on CodeProject like Pete O'Hanlon, who helped manage the sensors, this wouldn't have happened. The code I had written was garbage in comparison. Listening to other people is so important."