Ballastic Case Study

By Edward Correia

Introduction

In October 2012, Matthew Pilz, a programmer from rural Wisconsin, accepted a challenge. He had been lurking around the CodeProject website doing research when he came across a blog post by Lee Bamber, programming legend and founder of TheGameCreators.com, urging people to take part in the Intel® App Innovation Contest. Using a game idea that had stuck with him from years earlier, Pilz entered the contest, and in two months submitted his app, Ballastic. It won the gaming category's top prize.

About the Ballastic App

Ballastic was inspired by CANO-Lab's *Pendulumania**, a Japanese freeware game that Pilz had played in the late 1990s. The object of the game was to collect sparkling targets for points. Ballastic expands on that idea by adding rewards for capturing certain targets and for avoiding hazards and challenges that increase in number along with the difficulty levels. “I thought that would be the perfect platform using the touch interface to create the vision in my head,” Pilz said.

Figure 1. *As Ballastic game levels advance, balls are added to the game piece to increase the difficulty of movement.*

In addition to participating in the competition, Pilz said the original goal of Ballastic was to make a casual game that could easily be learned and played by people at any age or skill level. "I didn't have any specific target demographics. Simplicity is my goal with every app I develop," said Pilz.

Mission accomplished. Pilz has received feedback from both kids and adults saying that Ballastic is a great game that they have a good time playing.

Figure 2. *The Ballastic start screen.*

Since this was the first application Pilz had developed for Intel's Ultrabook™ device platform, he wanted to take advantage of as many of the platform's unique capabilities as he could and use them in unique ways. As the game progresses, the backgrounds change and the difficulty increases, similar to Lima Sky's *Doodle Jump**, which also presents just a single level that gets harder and harder. Ballastic's backgrounds change in another unique way. "The light sensor capability of the Ultrabook device is one of the coolest features than I have seen on any development environment. That capability allowed me to actually create different themes and different graphics depending on the level of light in the room."

By using the ambient light sensor in this way, Pilz was able to program the game to take on different appearances based on where it's being played. If the game is played outside in the sunlight, it will have a contrasted appearance, compared to being played in a room at night where it will have a totally different look. And for indoor play, backgrounds can vary along with screen brightness based on how much light is in the room. "I thought that would be an interesting concept," said Pilz.

Pilz already had experience with touch, having built several apps for the Apple iPad*, iPhone*, and other handheld and mobile devices. But he was naturally excited to have access to the increased computing resources afforded by the Ultrabook device platform. "Obviously the increased hardware support and graphical capabilities of Ultrabook provide an environment for more graphically rich and CPU-intensive applications than what you can get currently on mobile devices."

The increased processing power and feature set of the Ultrabook device allowed Pilz to add more shine and polish to Ballastic than could have been easily achieved on mobile devices. The high

resolution and wide-screen display allowed him to design assets natively for 1600x900 and higher resolutions at a 16:9 aspect ratio, whereas most mobile devices require that the game be 1024x768 or lower, and often at a 4:3 aspect ratio, thus diminishing the overall style and available game space. The graphical assets and sprite effects would likewise suffer from performance and visual discrepancies if translated to a less powerful platform. Most notably, the fluid background effects and dynamic particles seen throughout the game—comprising hundreds of sprites with constantly changing properties—would not perform as well on a mobile device without requiring significant optimizations and reduction in quantity.

Another benefit of the Ultrabook device is that its GPU supports a much higher refresh rate than the 30 to 60 frames per second that can be achieved on most other platforms. “Although a lower frame rate is certainly acceptable, being able to uncap Ballastic so that it runs at several hundred frames per second on the Ultrabook allowed me to provide exceptionally smooth and responsive controls and animations throughout the game,” said Pilz.

The Challenges of Creating Ballastic

Pilz faced several major challenges when developing Ballastic for the Ultrabook device, including the following:

- A small window of time in which to develop, test, and submit the app
- An unfamiliarity with the Ultrabook device platform
- Adapting to differences in touch development between iOS and Windows* 8

Developing, Testing, and Submitting the App

Pilz started in mid-October, about one month before the November 20 deadline, so he spent about a month-and-a-half of full-time, solid work—including several sleepless nights—to get his app to where he felt comfortable releasing it. Pilz had made an early design decision to make the app touch compatible, so its buttons and controls had to be large enough to allow users to touch them without a problem. He built an on-screen keyboard for entering high scores, and everything within the game can be accessed purely by touch. Pilz also incorporated the Ultrabook device keyboard and mouse as optional inputs for gameplay and menu navigation.

All-in-all, Pilz felt the experience of implementing touch controls on Windows 8 was similar to that of iOS, but there were a few bumps. “The biggest obstacles I found were design-oriented instead of SDK- or programming-related,” he said. One challenge was to ensure that all functionality could be interacted with easily and intuitively via touch. This meant bigger buttons and icons than what would be necessary if supporting only mouse input. The touch sensor functions were abstracted out nicely using the App Game Kit* (AGK), a set of OpenGL*-based libraries that made it a breeze to implement without having to dive into any of the raw SDK commands for such functionality.

Although Ballastic uses only single-touch controls, Pilz said it would have been easy to support multi-touch if the game needed to. AGK includes functions such as GetRawTouchCount(),

GetRawTouchCurrentX(index), GetRawTouchCurrentY(index), GetRawTouchLastX(index), and GetRawTouchLastY(index) that can interpret as many touches as the device supports.

Pilz said the native SDK commands provided by Intel also were straightforward. Intel's SDK uses common touch interfaces provided by Microsoft, including three different mechanisms for handling touch input: WM_TOUCH for supporting Windows 7 and 8 Desktop; WM_POINTER, which is specific to Windows 8 Desktop; and PointerPoint for developing a Windows 8 Modern UI (also known as "Metro") apps. Ballastic was developed as a Windows 7/8 desktop application, so had it been developed using the standard interfaces it would be most closely associated with WM_TOUCH events. This aspect allows it to run on any modern Windows desktop machine, supporting a greater number of devices than the other two methods. Much like iOS, Windows applications generally store touch information in a simple array or list that can be iterated through to read and interpret the touch points with ease.

The Ultrabook™ Device Platform

Fortunately for Pilz, Lee Bamber and the people at TheGameCreators.com, of which Pilz has been an active member since its inception in 1999, were hard at work on the latest beta of AGK, which does most of the game programming's heavy lifting. Pilz followed Bamber's blog from Intel's Ultimate Coder Challenge and read that he was adding features to support the various Ultrabook device sensors in beta versions of AGK, which he subsequently released to the community. Pilz knew he had to come up with the best platform and language to develop a game quickly. Bamber's code mapped Ultrabook device functions to a few easy-to-understand functions in both Tier 1 (BASIC) and Tier 2 (C++):

```
Sensor Support

GetNFCExists()
GetGeolocationExists()
GetCompassExists()
GetGyrometerExists()
GetInclinometerExists()
GetLightSensorExists()
GetOrientationSensorExists()


Notifications

NotificationCreate()
NotificationReset()
GetNotification()
GetNotificationData()
GetNotificationType()
SetNotificationImage()
SetNotificationText()

Near Field Communications (NFC)

GetRawNFCCount()
GetRawFirstNFCDevice()
```

GetRawNextNFCDevice()
GetRawNFCName()
SendRawNFCData()
GetRawNFCDataState()
GetRawNFCData ()

Geolocation

GetRawGeoLatitude()
GetRawGeoLongitude()
GetRawGeoCity()
GetRawGeoCountry()
GetRawGeoPostalCode()
GetRawGeoState()

Compass

GetRawCompassNorth()

Gyrometer

GetRawGyroVelocityX()
GetRawGyroVelocityY()
GetRawGyroVelocityZ()

Inclinometer

GetRawInclinoPitch()
GetRawInclinoRoll()
GetRawInclinoYaw()

Ambient Light Sensor

GetRawLightLevel()

Device Orientation Sensor

GetRawOrientationX()
GetRawOrientationY()
GetRawOrientationZ()
GetRawOrientationW()

Touch Sensor

GetRawTouchCount()
GetRawTouchCurrentX()
GetRawTouchCurrentY()
GetRawTouchLastX()
GetRawTouchLastY()
GetRawTouchReleased()
GetRawTouchStartX()
GetRawTouchStartY()
GetRawTouchTime()
GetRawTouchType()

```
GetRawTouchValue()
```

Complete documentation of the above commands will be available on AppGameKit.com once the final version of AGK 1.08 is released, as well as within the beta downloads for existing customers.

For Pilz, this meant that hardware features of the Ultrabook device platform could be tapped using a few simple functions. Best of all, the AGK includes BASIC and native-language support. "Those looking for more power and functionality than the standalone AGK BASIC language can provide are free to create AGK applications using Visual Studio*, Xcode*, Pascal, Eclipse*, and other environments and languages," wrote Pilz on his CodeProject page. He also noted that core commands of the AGK can easily be translated between Tier-1 BASIC and the Tier-2 native languages.

AGK saved Pilz an enormous amount of time, and he credits Bamber for the success of his project. Because the toolkit's developers abstracted out all of the Ultrabook device sensors and made them into a high-level API, Pilz was able to create rapid prototypes. He was also able to just call the different commands and read the values from the numerous sensors, including the ambient light level, accelerometer, and touch events, without having to spend days or weeks manually coding in the C++ environment or whatever the native SDK required at that time.

However, the first few weeks were anything but smooth. "It was nerve-racking because AGK is a closed source library and it wasn't fully prepared for Ultrabook support when the competition began," said Pilz. "The AGK developers consistently released new beta builds throughout the competition—sometimes several iterations in a single week—each with increasingly comprehensive support for Ultrabook. Eventually AGK had every feature of the Ultrabook available via simple function calls." Unbelievably, the final beta came out the day before the competition ended. So Pilz waited until that point to make sure that everything was supported and working on the Ultrabook device before he submitted the app to the Intel AppUp® center. Pilz credits the guys at The Game Creators for working around the clock to make sure this application development kit was compatible with the Ultrabook device. Good thing, too, because additional challenges arose that caused the project to lose time.

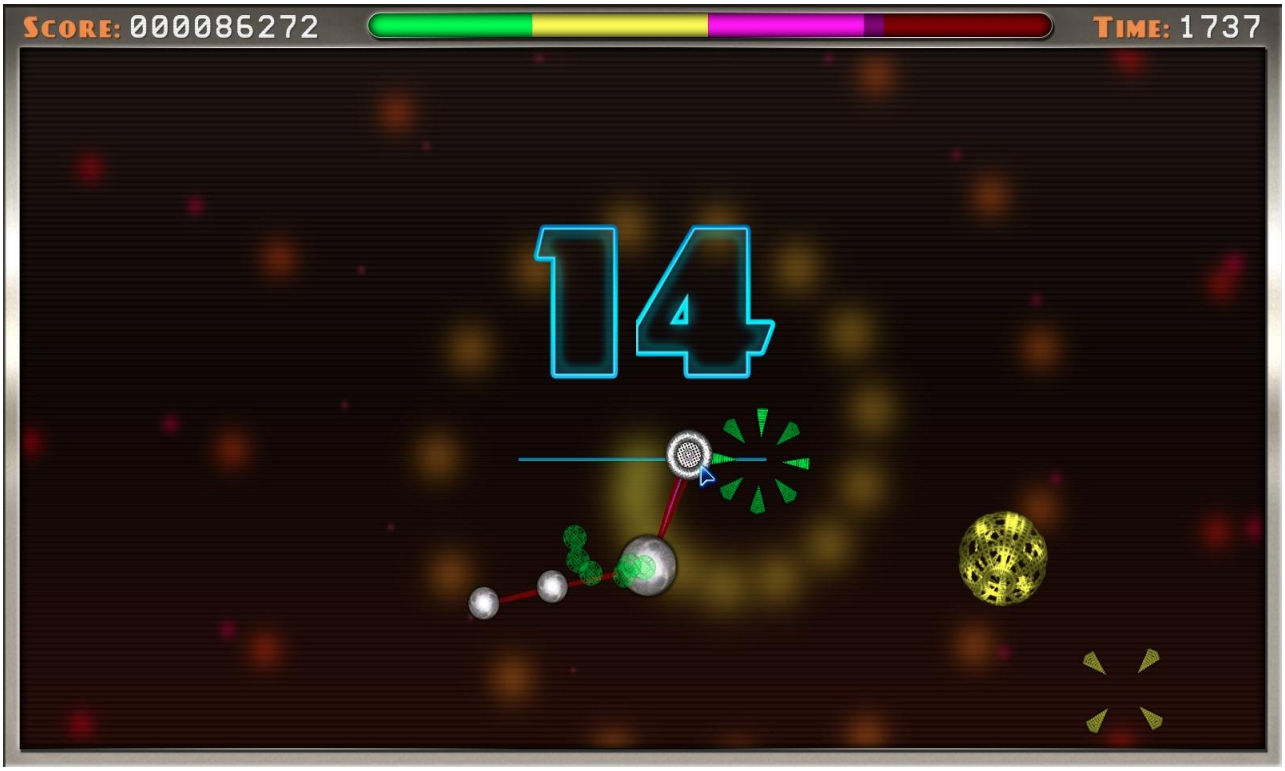

Figure 3. *As game play advances and objects are captured, the ball gets bigger and heavier, making it increasingly difficult for the player to maneuver and putting ever more stress on the elastic band that holds the balls together.*

Development Contest Rules

In addition to several snags with the Comodo certificate issuing system, Pilz hit a wall when submitting his app to the Intel AppUp center, Intel's Ultrabook device app store. Pilz warns other developers to be sure to dot all i's and cross all t's in their application submissions, including the meta data, as Pilz had his application rejected at the 11th hour when he did not include the registered trademark symbol next to the "Ultrabook" product name in his game description. Pilz was not the only applicant to run into that hurdle, but thankfully Intel was quick to approve the app once this minor detail was corrected.

Testing for Ultrabook Devices

With the technicalities behind him, Pilz turned his focus on testing and on acquiring the proper tools in the development kit to interface with the Ultrabook device. Having an Ultrabook device himself, he was able to test the sensors that he wanted his app to use.

The experience overall was a positive one, and Pilz said he'd be glad to do it all again and continue developing apps for the Ultrabook device. "Absolutely, yes. It was a fun and unique experience for me, even having developed for the iPad and Android* and other devices. I just think having that raw, high-end computer power and a good processor and video card [lets people] develop apps that you can't really create on most mobile devices. Even though they're getting more powerful, they still don't compare to a dedicated Ultrabook."

It's Not about the Money

While Ballastic is a free app, Pilz does generate revenue from other applications he's built with the company he founded, LinkedPIXEL. In the future, Pilz might switch to a free/premium model, offering a free limited version with an upgrade to remove ads or purchase new levels. "My game lends itself well to limitless new levels and power-up expansions, but [charging for it] hasn't crossed my mind too much."

For developers building apps for today's touch screen mobile devices, Ultrabook devices, and particularly running Windows 8, Pilz believes it's important to think outside of the box when developing these new-style apps. "If you're a traditional desktop programmer, you might build an interface that works well with keyboards and mice, but we also need to consider how people might benefit from a slightly different design to make things more efficient for those who may be using alternate input methods." Pilz also points users to resources such as CodeProject and the Intel® Developer Zone, and, of course, libraries like AGK, and feels that developers should still become familiar with Intel's SDKs because it's good to know exactly what features and commands the Ultrabook device supports.

Pilz also recommends developing code with a mind toward reuse, ensuring that interfaces and input controls easily adapt in the future, because of the new types of input technologies that will be available. Developing an interface that actually works well with current technology and is also adaptable will open up the doors to a much wider range of users.

Helpful Resources

Pilz relied heavily on the App Game Kit platforms and features for source code libraries developed specifically for the Ultrabook device, its sensors and interfaces. He was originally led to AGK and the App Innovation Contest through contacts made on CodeProject, a collaboration web site for developers. Pilz also made regular use of the Intel® Developer Zone for reference materials related to Intel SDKs.

About Matthew Pilz

Matthew Pilz spent his early years in rural Wisconsin, enthusiastic about computers and game development. It all started when his brother got a Commodore 64, and Pilz has been working with computers ever since. With an associate's degree in E-commerce and web administration from Milwaukee Technical College, a Bachelor of Science in Web Technologies from Bellevue University, and various computer-related technical certificates, Pilz has spent the last decade focused on web design and application development for a variety of platforms. In spring of 2013, Pilz also won a grand prize in the Intel® Perceptual Computing Challenge by creating a prototype application, Magic Doodle Pad, using a perceptual camera and the Intel® Perceptual Computing SDK.