Don't Get Bit: Avoiding the   
Windows Phone Watchdog

By Cullen Waters, Software Development Engineer

Advanced Technology Group (ATG), Microsoft

Failing to implement the best practices recommended in this white paper may result in your application being shut down automatically by the Windows Phone Watchdog service even if your application runs fine on your current development devices. Watchdog is a service on Windows Phone that monitors unresponsive applications.  Unresponsive applications are put into the background and the user is taken back to the Home screen. Eventually the application is killed and many users are likely to interpret this as the application crashing at launch.

The system defines unresponsive applications as ones that do not present information to the user within 10 seconds of startup. Your application has approximately 9 seconds to startup, to process framework code, and to draw it to the screen.

The approach that works best for Silverlight applications is to include an image named SplashScreenImage.jpg in the root of your application's XAP file.  Silverlight's application framework will draw this image to the screen on your behalf, and will take care of keeping the Watchdog at bay.

For XNA titles, things are not so simple.  You can include a SplashScreenImage.jpg in your project, but because of technical complications with the application model's interaction with Silverlight, that image will only display on-screen for a fraction of a second. For this reason, we recommend that the default splash screen image not be included in an XNA title.

It is content loading that prevents an XNA title from drawing to the screen at game startup.  Many games need to load quite a bit of content before anything actionable can be displayed to the user.  There are a number of ways to avoid the Watchdog while still loading a large amount of content.

Initially, it was believed that Watchdog could be avoided by clearing the graphics device during initialization of an XNA title. This approach was abandoned because it does not work correctly in all cases. There are differences in the transitions in the XNA Framework when resuming from a tombstone that prevent this approach from succeeding.

If your title uses the GameStateManagement framework, or a similar game state system, you can add a new game state for showing a splash screen. This game state would load the bare minimum needed to draw a splash screen (usually just a single texture), and then would continue the full content load in the background.  The code to do this is dependent on the game state system your title uses. The key point is to load enough content to display the splash screen, and then to display that content on the screen as quickly as possible.

This system could be expanded to one that allows multiple splash screens to be loaded and displayed one after another.  This would be very similar to what you see in many commercial Xbox 360 titles.

To make the experience more immersive, you could author animations into your splash screens, so that they rotate or pulsate, or use some other interesting effect.  As you travel down this road, remember the Watchdog.  It is easy to go overboard with your splash screens, and end where you started: on the Home screen with the user believing your game just crashed.