Increase and Optimize UITableView Performance

In this chapter, you will do the following:

- Use the benchmark tools you saw in Chapter 2 in a real example.
- Work step by step with an example of scrolling performance optimization.
- Use techniques to optimize UITableView performance.
  - Basic techniques to optimize simple and basic cells in UITableView.
  - Hardcore techniques using code to actually draw the view inside the cell.
  - Basic optimization techniques for cells need animation like editing and reordering.
  - Other basic techniques that developers need to know.

iPhone apps usually display data in the list format. Apple provides excellent tools for basic developers: UITableView and UITableViewCell. If developers only want to use basic features to display a small image on the left and text in the center, the default Apple control should work well. However, when you have to do lots of customization to the listing order to have two or three images and text in different places, you’ll run into problems. Sooner or later, you’ll run into performance problems with jerky scrolling of the UITableView, especially on old devices like iPhone 3G.

Introduction to the Examples

For this exercise, I will measure the performance based on two main factors: the speed the tableView dequeues, creates a new cell, or returns the cell back for the operating
system (OS); and how fast the OS can render your cell to display it to the machine. The first one can be measured by NSLog easily enough; the second one is more complicated and can only be measured by CoreAnimation.

To demonstrate, I will walk you through two different examples. One just contains an avatar and a textblock; the other one has complicated views with many subviews inside. By going through these two examples, you will see many different ways to optimize the scrolling performance of a UITableView.

At the end of this chapter, I will also list other important points that I don’t have time to cover in detail in this chapter. These are not usual mistakes, but if a developer is careless enough to make one of these mistakes, it will cost him a whole day of benchmarking and testing to figure out the problem. I want to make sure that you have enough skills and knowledge under your belt to handle any situation.

Sometimes, optimization is so simple that it only requires a few changes here and there inside the code. However, in other cases, like in the second example, optimization will require rewriting the whole code base to fit into a better, more optimized model. I hope that after I go over the examples, you will have a clear idea of the program’s structure so you can make the right decisions at the beginning and will not need to rewrite the whole code base.

### Reviewing the Instrument Tool

For this chapter, you will use the CoreAnimation tool to benchmark the render performance of iPhone OS. This will help you to know if the problem is within the computing process or in the displaying process. Chapter 2 covered this tool, so this chapter only provides a brief review.

Figure 3–1 shows the main view of the CoreAnimation tool with the three parts that you need to look at when running the tool and Figure 3–2 shows the performance readings.

![Figure 3–1. Main parts in the CoreAnimation tool](image-url)