ART- Improving Execution Time for Flash Applications

Ming Ying, University of Alberta, Canada
James Miller, University of Alberta, Canada

ABSTRACT

Rich Internet Applications (RIA) require fast execution time and allow richer, faster, and more interactive experiences. Flash is a common technology for building RIAs. Flash programmers usually specialize in graphic design rather than programming. In addition, the tight schedule of projects makes the Flash programmers ignore non-functional characteristics such as the efficiency of their systems; yet, to enhance Flash users’ experiences, writing efficient ActionScript code is a key requirement. Flash programmers require automated support to assist with this key requirement. This paper proposes a refactoring system called ART (ActionScript Refactoring Tool) to provide automatic support for Flash programmers by rewriting their ActionScript code to make their applications faster.

Keywords: ActionScript, Efficiency, Refactoring, RIA, Web 2.0

INTRODUCTION

Nowadays, the Web 2.0 has significantly changed the way users interact with the web. The nature of Web 2.0 requires the growth of Rich Internet Application (RIA) technologies, such as Flash and Ajax. RIA technologies allow richer, faster and more interactive experiences. It breaks the old click-and-wait user experience mode. Instead, by only changing updated information without refreshing the entire page, RIA makes the web-based application feels more like a desktop application (Hewlett-Packard, 2007).

Adobe Flash is a multimedia platform for creating interactive and animated web sites. Flash movies and games are commonly integrated into web pages as components for entertainment or advertisement. For example, there are many Flash games in Facebook (n.d.). Flash contains ActionScript 3.0 (AS3) which is an object-oriented scripting language based on ECMAScript. To view these Flash movies and to execute these ActionScript files, browsers require the Adobe Flash Player add-on. Adobe (2009) claims that about 99% of Internet-enabled desktops have the Adobe Flash Player (10-current version) installed. Many smartphones including Android, BlackBerry, Palm webOS devices, etc now also support Flash Player 10.

The performance of Flash applications, especially in mobile devices, directly influences the user’s experience. Speed is one of the most important requirements for mobile devices’ us-
ers (Buyukozkan, 2009). The user’s experience of Flash applications is partially determined by the download time and execution time of Flash files. Download time depends on the size of the Flash file and the connection speed to Internet. The file size can be reduced through the compression of the file. Execution time relies on the processing power of the client machine and the performance of the ActionScript code. Although reducing Flash files’ size is helpful, it is not the key point; writing faster and more efficient ActionScript code is the most useful way to improve the user’s experience.

The quality of Flash code is highly dependent on the developers; however, Flash programmers often “have backgrounds in music, art, business, philosophy, or just about anything other than programming. This diversity results in awesome creativity and content” (Skinner, 2007), but imposes technical challenges. In addition, the tight schedule of a project tends to result in developers concentrating on “getting the functionality correct” (Skinner, 2007), while ignoring non-functional characteristics such as efficiency.

This paper presents a “refactoring for efficiency” Flash support system, ART (ActionScript Refactoring Tool), to help AS3 programmers produce more efficient code by automatically transforming their ActionScript code. This paper is organized as follows: First we discuss the efficiency problems in traditional web applications and RIAs, and related works. Then we introduce the Flash execution model. Next we analyze possible strategies for improving efficiency in Flash applications and then describe the design of ART. Finally we provide an evaluation of our system and conclude the article.

EFFICIENCY PROBLEMS IN TRADITIONAL WEB APPLICATIONS AND RIAS

Problems

In traditional web applications, the content of the web pages is commonly text and simple graphics like images, which do not consume many CPU resources. As a result, efficiency problems in traditional web applications are not a big problem.

However, with the advent of the RIAs, users have higher expectations on web applications. The main difference between RIAs and traditional web applications is the level of interaction. The interactions of traditional web applications are restricted to the operations with visual objects, such as document objects, frame objects and button objects. RIA technologies emphasize on the rich and engaging user experience. Graphics, animations and different visual effects are used to create highly dynamic, interactive web pages. Speed becomes an important requirement for the new generation of web applications as it directly affects users’ satisfaction. According to Card’s research (Card, Robertson, & Mackinlay, 1991):

1. Users feel that a system responses immediately if the response time is around 0.1 seconds;
2. Users feel that a delay has occurred if the response time is around 1 second; and
3. The limit for users to retain interest in a specific task is 10 seconds, after that, they give up waiting and turn to other tasks.

In RIA systems, these times can also include communication overheads. Though dynamic and interactive web pages are the great features of RIAs, they can give rise to serious performance problems. For example, Flash movies and games consist of numerous different graphics (vector and bitmap graphics) which are manipulated to provide a visual experience. After Flash movies or games have been downloaded to the user’s machine, these CPU-intensive tasks become the biggest bottleneck and are the principle source of performance problems. If the graphic objects are not well programmed and organized, it will lead to delay or even unresponsiveness. Therefore, RIA client-side technologies require efficient code, such as efficient ActionScript for Adobe Flash, JavaScript for Microsoft Silverlight and Java for
Cooperation Between Agents to Evolve Complete Programs
[www.igi-global.com/chapter/cooperation-between-agents-evolve-complete/24151?camid=4v1a](www.igi-global.com/chapter/cooperation-between-agents-evolve-complete/24151?camid=4v1a)

Risk-Based Privacy-Aware Information Disclosure
[www.igi-global.com/article/risk-based-privacy-aware-information-disclosure/136467?camid=4v1a](www.igi-global.com/article/risk-based-privacy-aware-information-disclosure/136467?camid=4v1a)