Expert Systems with Applications 42 (2015) 5245–5255

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Optimization model for web based multimodal interactive simulations

Tansel Halic [a,*], Woojin Ahn [b], Suvranu De [b]

[a] Computer Science Department, University of Central Arkansas, 201 Donaghey Ave., Conway, AR 72035, United States
[b] Department of Mechanical, Aerospace and Nuclear Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, United States

ARTICLE INFO

Article history:
Available online 5 March 2015

Keywords:
Interactive web environments for medicine
Optimization
Simulation
Virtual reality
Web-based interaction

ABSTRACT

This paper presents a technique for optimizing the performance of web based multimodal interactive simulations. For such applications where visual quality and the performance of simulations directly influence user experience, overloading of hardware resources may result in unsatisfactory reduction in the quality of the simulation and user satisfaction. However, optimization of simulation performance on individual hardware platforms is not practical. Hence, we present a mixed integer programming model to optimize the performance of graphical rendering and simulation performance while satisfying application specific constraints. Our approach includes three distinct phases: *identification*, *optimization* and *update*. In the *identification* phase, the computing and rendering capabilities of the client device are evaluated using an exploratory proxy code. This data is utilized in conjunction with user specified design requirements in the *optimization* phase to ensure best possible computational resource allocation. The optimum solution is used for rendering (e.g. texture size, canvas resolution) and simulation parameters (e.g. simulation domain) in the *update* phase. Test results are presented on multiple hardware platforms with diverse computing and graphics capabilities to demonstrate the effectiveness of our approach.

1. Introduction

Multimodal interactive simulations (MIS) are highly demanding in terms of computational resources. Such systems have wide applications in medical training, flight simulation, video gaming, movies and other industries. MIS systems especially virtual reality applications require multiple components to work synergistically to achieve a high degree of immersion. These may include realistic visual rendering engine to display photo-realistic images and physics engine to simulate real world phenomena including rigid body motion, deformation of solids, fluid flow, and multi-physics phenomenon. Multimodal interactions require interfaces such as mouse, trackballs, haptic and tracking devices. Enabling interactions with multiple hardware devices while maintaining expected realism for both visualization and simulation require complex algorithms and considerable CPU time. However, faster execution rate is necessary in applications where real-time interactivity is mandatory. This is especially challenging when multiple hardware platforms must be supported with diverse hardware specifications.

Unfortunately, conventional MIS systems are designed to work with specific software and hardware settings. Enabling cross-platform compatibility (Halic et al., 2011; Maciel et al., 2010) may not be the right approach as it requires *a priori* knowledge of the platform, intricate compilation and installation procedures (Engelke, Becker, Wuest, Keil, & Kuijper, 2013). This greatly limits accessibility, usage and portability. Besides, such an approach may not be feasible on proprietary target platforms. On the other hand, rapid growth of web and ubiquitous computing platforms such as tablets and smart phones have facilitated the accessibility, mobility and usage of applications regardless of the underlying platform. To access such advantages, applications should be platform independent and be accessible over the World Wide Web. The web introduces greater flexibility as the applications do not need to be installed as in the case of traditional native applications. MIS systems on the web also allow for low development and maintenance costs. This requires the use of open standards of the web (e.g. HTML5). Device anonymity in terms of hardware and the software needs to be maintained. Hence, migrating MIS to the web offers true portability and platform accessibility along with widespread usage.

The Software Framework for Multimodal Interactive Simulations (Π-SoFMIS) is a platform-independent framework designed to facilitate development of 3D interactive applications over the web using WebGL (Halic, Ahn, & De, in press; Halic, Ahn, & De, 2011). WebGL is a plug-in free JavaScript based visualization technology, released in 2011 (now part of HTML5 standard) allows for the development of realistic and interactive 3D applications on the web browsers (“WebGL Specification,

* Corresponding author.
E-mail addresses: tanselh@uca.edu (T. Halic), ahnw@rpi.edu (W. Ahn), des@rpi.edu (S. De).

http://dx.doi.org/10.1016/j.eswa.2015.02.026
0957-4174/

2012). Applications based on Π-SoFMIS can therefore operate on multiple web browsers running on a range of hardware platforms including desktops and mobile computing devices. The performance of visualizations and simulations depends heavily on the hardware platform, which significantly affects user experience. With hardware upgrade cycles becoming shorter, designing an application that can capitalize on the prospective upgrades becomes even more challenging.

Platform independent applications not only need to detect target hardware but also adjust their performance to future changes of hardware. This process is often more challenging than the development of the application itself. Hence, it is essential to develop a strategy that allows automatic detection of target hardware resources and optimize resources to run most efficiently on the device. In this paper, we address this issue by developing a strategy that is capable of computing visualization and simulation parameters based on the client hardware while respecting the constraints set by the application.

One common use of MIS systems is virtual reality (VR) based web applications (Arnab, Petridis, Dunwell, & de Freitas, 2011). In VR applications, the realism of the application is highly affected by the execution constraints. For instance, VR based simulation with haptic interface devices; force feedback computation needs to be performed at 1 kHz execution rate for smooth touch sensation. Visual rendering rates entails execution rate at least 30 Hz for real-time interactivity. Any failure to meet these execution constraints due to lack of optimization of the device computing resources drastically reduces the user experience, which causes disruption in the sense of immersion.

Our performance optimization framework models the performance of visualization and simulation using non-linear mixed integer programming (Floudas, 1995). This model satisfies the visualization and simulation performance constraints while maximizing the visual quality for a specific device. The optimization does not make any assumption on hardware capabilities. The device capabilities are parameterized and extracted in the identification phase, which respects device anonymity. The subsequent pre-optimization phase of the simulation is performed on the server side resulting in elimination of overhead on the client device. This makes the approach more suitable for low profile devices such as smart phones.

Our optimization model also takes into consideration the constraints set on the quality of visual rendering and performance of physical simulations at the application level. Factors such as load due to visualization and the programmability of GPU are also considered. Texture sizes, display resolution, canvas up-scaling ratio and geometry discretization level are determined as outputs. Computing the visualization and simulation parameters before the start of simulation prevents any unwanted effects during the application execution such as visual artifacts existing in previously proposed real-time approaches. Unlike existing approaches our model also accounts for run-time performance of simulation to avoid slowdowns, lag or freeze due to scarce CPU resources. Our optimization approach has been implemented and tested in the context of Π-SoFMIS. However, the methods used here may also be applicable to other software environments.

2. Literature review

Real time measurement and control of applications involving visualization and simulation is an on-going area of research. Tack, Morán, Lafruit, and Lauwereins (2004) proposed real-time control of applications involving 3D graphics content targeted for mobile terminals such as PDAs. Their aim was to decode different 3D content while considering the dynamic load on the device. They derived a heuristic algorithm to approximate the content to be rendered for a 3D scene. Their control algorithm estimates the rendering time by analytically approximating the rendering pipeline (“Mesa Home Page., 2012). Continuous processing power on the client and server sides is thus necessary. This type of analytical modeling assumes that the entire rendering pipeline is carried out on the CPU with a single thread, and is also not suitable for the estimation of execution time for current modern GPU-based systems which include multiple cores (hence multiple threads) and system on chip systems (SoC) (Wolf, Jerraya, & Martin, 2008) type architectures. Finally, in the current platforms separate acquisition of each parameter corresponding to rendering stages such as vertex and triangle clipping, rasterization of line, triangle and pixels is not practical due to lack of driver support at the application level.

In further work from the same group (Tack, Lafruit, Catthoor, & Lauwereins, 2005; Tack, Lafruit, Catthoor, & Lauwereins, 2006) they defined a cost function and used an unconstrained optimization model to increase visual quality. They performed computation of view dependent Pareto graphs offline and associated the graph for each object in the scene to change in the level of detail (LOD) during run-time. In their approach, optimization of the textures and artifacts due to level-of-detail changes were not considered. Wimmer and Wonka (2003) tried to estimate the rendering time with analytical approximation based on heuristics. The heuristics are based on triangle count, transformed vertex count, cost derived in Funkhouser and Séquin (1993) and estimation based on sum of the cost of total transformed vertex and total generated pixels. Their work focused on estimation of rendering time rather than a framework for optimization of performance on multiple hardware devices.

A signature-based profiling approach was presented in Mochocki, Lahiri, Cadambi, and Hu (2006) for estimation of rendering performance. In this approach, “signature” refers to information generated from modifying the rendering routine. Their idea was to collect and update information (signature) during rendering and estimate the performance of the next frame from previously collected signatures using a distance metric. The extensive data collection process requires modifications to the original application and native support from the hardware platform (e.g. hardware performance counters) to record accurate performance metrics, which may not be feasible for many applications. Their validation results were based on software simulation of ARM chips rather than actual physical devices. Their major goal was to predict workload to save energy by adjusting voltage/frequency scaling of the SoC chips instead of changing quality offline/online or performance of an application.

Wong and Wang (2008), Wong and Wang (2010) tried to estimate 3D rendering performance using a black box-type linear parametric model and neural networks. In their study, input parameters to the model such as parameters affecting the adaptive controller, rendering process, quality of service assurance are not explicitly conveyed. Their validation results were limited to desktop machines with no guarantee on the scalability to other systems such as mobile or tablet devices. The work did not describe the training process and calibration of the model and more importantly computation and memory overhead for devices that have minimal resources.

Ngoc, Lafruit, Deconinck, and Lauwereins (2003) proposed a middleware that decides allocation of the hardware and software resources for embedded systems. Their system targeted the field-programmable gate arrays (FPGA) architecture. The model proposed is for discrete set of computational tasks designated for batch processing. They did not consider hardware constraints such as memory. The set of applications that they aimed for are not clear from the work. Moreover, no validation studies were presented.