Mobile Simulation with applications for Serious Gaming

Andrew Jeffery, Jonathon Panke, Gabriel Wainer
Department of Systems and Computer Engineering
Carleton University
Ottawa, Canada
jeffery.andrew@gmail.com, jonathonpanke@cmail.carleton.ca, gwainer@sce.carleton.ca

Keywords: Discrete event simulation, DEVs, mobile devices, serious games

Abstract
Since simulation is normally required as the basis for serious gaming applications, using RISE simulation services a serious game application can be deployed on a mobile device using a properly developed mobile client. We propose a mobile client that leverages the cloud services provide by a RISE server to obtain simulation data to be used in a serious gaming application. In particular, our prototype will provide a stock trading game which will be based on a Brownian motion economic model that will be analyzed comparatively to real data. The merits of the mobile client created as a serious game will also be discussed.

1. INTRODUCTION

With the growth of network connectivity in past years there has been a dramatic increase in the applications and services that are provided for mobile devices [1]. Additionally to overcome the hardware processing limitations of current technology new methodologies and design architectures have emerged, including distributed systems and cloud services. Such technologies make it possible to bring complex, process heavy simulation to a mobile device. As further developments in mobile communications arise new tools are being deployed on mobile platforms. When taking into the consideration the increasing popularity of mobile tools and applications in younger generations, there is a large demand to try to incorporate mobile devices in teaching practices. It has been suggested that the best way to do this is using MSG (Mobile Serious Games) game as explained in [2].

The idea of serious gaming is to provide an environment analogous to a reality. In this environment a problem is presented to the participants which they must solve using similar techniques and knowledge that would be required if the event occurring was real. This allows for the development and practice of required skills before they are required in the field.

![Figure 1: Aspects that make a serious game. [3]](image)

To successfully design a serious game many aspects are required. One of the largest aspects of a serious game is simulation, which the goal of the simulation is to emulate a real life environment that follows specific rules that related to the real world. The mixture of simulation and learning within the format of a game is shown in Figure 1, in game based learning game is the region that incorporates a true serious game.

Applying the idea of serious gaming to a video game normally requires some level of simulation from the application in order to obtain the proper data to events that are occurring in the virtual system. Web-based simulation has become a popular technique to conduct online simulations. The idea is to use a Web server to implement the networked architecture using a particular architecture (client-server, High Level Architecture, CORBA, RPC, etc) [4]. Many advanced implementations have been built on SOAP Web Services to communicate [5] [6]. These simulation...
middle wares are complex to interoperate, and their composition scalability is limited. Instead, RESTful Web Services can solve these issues by imitating the Web interoperability style. Instead, the Representational State Transfer style (REST) focuses on the resources more than on the operations, solving the interoperability limitation and making easy the development of Mash-Ups [7] (which reuse and combine existing services to build a new web application).

Since this type of simulation can be computational heavy, these types of games traditionally have been limited to hardware not found in mobile devices. Yet with the ability developments in cloud computing and services there now exists systems that can provide simulation to a remote user as a cloud or web service. The RISE simulation environment provides an API to use its cloud services using RESTful commands to run simulation models remotely and retrieve the simulation results. In this article we present a mobile Smartphone client that uses simulation of a Stock Market environment that uses the inclusion of player driven missions to learn about the basics of stock trading. The client titled Stock Market Tycoon was originally developed with the desire to bring simulation data done on a distributed server to mobile devices. This later turned into a MSG with the goal of having mission for the user to achieve. The successful development of a client that meets the goals of allowing a user to have access to a simulation model remote and the results based on dynamically varies input values was achieved.

2. BACKGROUND

Serious gaming has evolved from what it used to be with war games, and board games. It has entered into the digital world, which has led to techniques in designing serious games. Today serious games for the financial stock market are very scarce. A few examples of modern games with the stock market are: Wall Street Survivor, MarketWatch, and UpDown [8] [9] [10]. All three of these web browser based games use real stock values in trying to teach their users about investing. These differ from our application greatly due to their lack of simulation, and required connection to the Internet at all times. Another example of similar work is found in [11] which demonstrates a simulation engine for a stock options game. Ours differs greatly due to ours being a serious game, and the patented engine is for a normal game which not based on the actual economic behavior of real stocks. Another difference is our focus on presenting the simulation on a mobile device, simulation on a mobile device in this way is discussed in [12]. Our client focuses more on using the application as a serious game and allowing the use of dynamic data for the simulations.

RESTful interoperability simulation environment (RISE) is a Carleton University simulation middleware. The goal of the distributed simulation middleware is to interlace different simulation environments to allow synchronization for a simulation, which is ran across a distributed network [13]. The goal of using the RISE server architecture in the project is to allow for cloud computing services, and allowing multiple users able to run simulations at the same time.

To use the RISE server an API is provided which uses REST commands. Table 1 is a list of the commands that will be used in the project.

<table>
<thead>
<tr>
<th>Action</th>
<th>Channel</th>
<th>HTTP Success Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Framework</td>
<td>PUT</td>
<td>201 (Created)</td>
<td>XML</td>
</tr>
<tr>
<td>Configure Framework</td>
<td>PUT</td>
<td>200 (OK)</td>
<td>XML</td>
</tr>
<tr>
<td>Submit Model</td>
<td>POST</td>
<td>200 (OK)</td>
<td>.zip file</td>
</tr>
<tr>
<td>Delete</td>
<td>DELETE</td>
<td>200 (OK)</td>
<td>None</td>
</tr>
<tr>
<td>Start</td>
<td>PUT</td>
<td>202 (Accepted)</td>
<td>None</td>
</tr>
<tr>
<td>Stop/Abort Simulation</td>
<td>DELETE</td>
<td>200 (OK)</td>
<td>None</td>
</tr>
<tr>
<td>Check Simulation</td>
<td>GET</td>
<td>200 (OK)</td>
<td>XML</td>
</tr>
<tr>
<td>Download Results</td>
<td>GET</td>
<td>200 (OK)</td>
<td>.zip file</td>
</tr>
</tbody>
</table>

Table 1: REST API with Messaging [14]

RISE itself uses Discrete-event systems (DEVS) models created with CD++ to run the simulations. DEVS is a formalism used in the modeling of both continuous and discrete worlds. Thus DEVS is an excellent tool to model a simulation environment. DEVS can be used to model a system whose states change based on the expiration of a time delay, or due to an input event, which with a stock market environment encompasses both these. [15] An input event for a stock market would be a random event that affects how a stock changes.

DEVS models are a simplified structure of what it represents in reality. The models are built upon the experimental expectations of the system; these constraints being the working conditions and its application domain. These constraints are a composition of atomic or coupled components.
Coupled models are just a collection of atomic models or coupled sub models. This allows for the creation of very complex and exquisite simulation models.

CD++ is a toolkit, which was developed for the use of modeling simulation environments using DEVS formalisms. It allows for Discrete Event Modeling which is derived from system theory and allows a system to be decomposed into sub-models. This feature of CD++ allows for a modular approach to developing simulation models, and due to the system theory principles creates reusable models. CD++ has support for creating two types of models, behavioral and structural models.

Behavioral Models are referred to as atomic models and consist of a single model. Cd++ allows for multiple atomic models to be connected creating more complex structural models. The connection of atomic models is done through ports; these ports exist for any atomic model and are used for passing data into models via input ports and receiving data generated by the model via output ports. Therefore a system can be a single atomic model or many atomic models; a collection of multiple atomic models is referred to as a coupled model or structural model.

Since a system is decomposed into some number of modular sub-systems, this makes interconnecting models easy and provides communication of data from model to model. This also allows models to be developed in iterations since additional behavior can be added in the form of an additional module (i.e. another atomic model). Models can be written in C++ allowing any C++ functionality to be implemented in a model, this yields flexible models and C++ has one order of magnitude faster execution then Java, which is another popular language to implement simulation tools. [15] A bonus feature of using RISE is that the user spaces on the servers are broken up into domains and frameworks, where there is no interference or shortage of resources between them. This is very useful because it allows for multiple users to have their own unique data which can’t be tampered with by other users. If many users are requesting simulations to be executed at the same time, each of them doesn’t have to wait for someone to free up resources.

The objective of RISE is to allow a plug-and-play interoperability paradigm for simulation tools. The way it achieves this goal, is decoupling the services from the required formalism. In other words, it provides a general purpose API to interface different simulators. In this way, the clients can use an experimental framework with a number of instances with different settings. This approach allows two or more instances to exchange messages with each other. One of the driving ideas of this system is to make every experiment persistent and repeatable, by storing simulation configurations. RISE exposes the tools through a set of URIs, so they are dynamically accessible from anywhere. When a client makes a request, it uses a specific URI. RISE looks for the resource, and sends the request to it, collecting and forwarding the response to the client. Clients connect to the URIs via HTTP channels. They use GET operations to read resource data, PUT operations to create new resources or update them, and DELETE to remove a resource. For example, a simple session could include the following sequence:

1. PUT http://.../cdpp/workspaces/bob/DCDpp /model, to create a model in the workspace “bob”;
2. POST http://.../cdpp/workspaces/bob/DCDpp /model?zdir=files, to submit the model files;
3. PUT http://.../cdpp/workspaces/bob/DCDpp /model/simulation, to start the simulation;
4. GET http://.../cdpp/workspaces/bob/DCDpp /model/results, to download the results

A complete description of RISE can be found in [16] and [17]. Although RISE can manage different simulators, we have a using CD++, a simulation toolkit based on the DEVS and Cell-DEVS formalisms. More information about it can be found in [18]. The most common way to use a simulator remotely is by designing a web-based tool. Byrne, Heaveya and Byrne in [19] provide an extensive review of web-based simulators, discussing the advantages and disadvantages of the client/server pattern applied to the simulation, and analyzing how and with which technologies it can be implemented. In the light of their paper, our work uses a hybrid simulation and visualization approach to drive a remote managed simulation (the client grabs the raw data and visualizes them).

3. SIMULATING BROWNIAN MOTION

Simulating a stock market environment can be achieved in many different ways. One of the most common approaches is using Brownian motion, in which Brownian motion is an introductory method to calculating trends in assets. The mobile application prototype we have built uses Brownian motion, and more specifically deterministic Brownian. The deterministic approach allows for the prediction of the assets in a manner that doesn’t allow for risk. The formula used for our model is [20]:

$$\frac{\Delta S}{\Delta t} = \mu S_t$$
\[ S_T = S_0 + rS_0 \Delta t \]

Where the variables are as followed:
- \( St \) = New Stock Price
- \( S_0 \) = Initial Stock Price
- \( r \) = Random Normalized Variable
- \( \Delta t \) = Difference in Time between \( S_T \) and \( S_0 \) as a fraction of a year

To show the realism in a Brownian motion simulated stock environment, simulations over a large time frame were done and compared to the real world environment. The real world stock example chosen to compare to was the well-known company BlackBerry (BBRY). The graph in figure 2 is the daily stock graph for March 20th 2013. Something to note while looking at the stock graph is that the market opens 9:30 EST and closes at 4:00 EST, and the graph plots the points in 5 minute intervals. Figure 2 shows the daily progress of the BBRY stock. The graph shows the BBRY stock opening with a price of $16.35 a stock and closing with a price of $16.53. The letters on the stock graph are news events that correlate with the BBRY stock for that day.

To test and validate the effectiveness of the deterministic Brownian motion DEVS model that was implemented two variables had to be calculated – the change in time and the number of data points for a given day. For this we got a value of \( \Delta t \) to be \(~0.0042\) which roughly equates to a 5 minute interval of time in accordance to a year, and 84 data points to mimic a normal stock day. Using these numbers to generate 5 trials, the simulation results can be seen in figure 3.

To further prove the validity of the simulation the 10% and 5% confidence interval of each simulation was calculated to give the following results. From these results seen in table 2, we can determine that the results can pass a confidence level test with above average results.

4. IMPLEMENTATION

The application is based on server-client architecture, and the vital piece of any server-client architecture is the communication protocol implemented that handles all the messaging and communication between the two entities. The implemented communication module in this project is the communication link for the mobile client and the RISE server, and handles all messaging between the two entities. The communication module has a number of responsibilities that assist that client aside from communicating with the RISE server.
Each state of the communication state machine represents a stage in the communication protocol that the client and server execute in order for new data to be obtained by the client. The data returned is dependent on the chosen model of simulation, and the client determines which model it is using before communication begins so that it can signal the RISE server to setup the appropriate framework.

There are four states to the communication state machine. The Ready state is the initial state when the state machine is instantiated. In the Ready state there is no activity occurring on either the client or server side; it is from this state that any communication begins. The Setup state indicates the framework has been initialized with the selected model and has pending data for a simulation. The Running state indicates no other data requests can be sent, since a simulation is currently running. The Done state indicates that there is pending simulation data that needs to be collected by the client.

5. APPLICATION AS A SERIOUS GAME

The core goal of this project was to bring the ability to run remotely simulations on the RISE server located on the Carleton University campus. This goal would allow a mobile phone the ability to run complex, process and CPU intense simulations by using the cloud services provided by the RISE API. The project was successful in meeting this goal by implementing a Windows Phone Client application that was able to distribute the more CPU intense task of simulation to the RISE server for the purposes of simulation a stock market environment. This data was sent using the REST commands that where specified by the RISE server documentation via compressed zip files. The Mobile client was then able to retrieve the simulation data and implement it as the basis for a stock market trading simulator called “Stock Market Tycoon”.
Stock Market Tycoon used the simulated data to present to the user the options of trading stock for profit and browsing a small selection of stocks that were present in a traditional stock market format. This format and the use of the simulation data provides a recreation of the same environment that is present in a real stock market aside for the real-time aspect since time progression of the market is at the user's discretion. Therefore the client application implements the target attributes of a serious game due to it's the recreation of a real system.

6. CONCLUSION AND FUTURE WORK

While the core goal of this project was met, there was a secondary focus that the team decided to focus on in additional to the stated project goals. This was the goal of a favorable user experience. Some features, that while implemented with consideration toward the user, were the user driven time advancement, the color coding of data on the GUI to represent a loss(red) or a gain(green) and various others.

Yet despite these considerations time restricted the implementation of all proposed ideas. Some features that could be implemented with further development could be:

- Shortcuts between the Portfolio, Stocks, and Missions screens.
- A better time scale for the stocks graphing feature.
- Improve the user experience for some of the more cluttered screens.

In a game play aspect, addition of computer players for a more competitive environment and additional more complex missions could give the application more replay value. The addition of more stock market trading options, such as short, and time advancing further than an hour, would also diversify the user's game play experience.

References


