Multi-Device Multi-player Gaming: Issues and Application

Danny C. Cheng, Allan Christopher S. Chu
Assistant Professor College of Computer Studies
De La Salle University
danny@ccs.dlsu.edu.ph, allan@ccs.dlsu.edu.ph

Abstract

The current advances in mobile technology, has made it possible for traditional pc games to be ported to the mobile device. The development of multi-user games on mobile devices has presented many challenges and issues. This paper presents the different issues that must be tackled when creating a viable multi-user, multi-device game. We tackled issues ranging from user interaction issues, graphics quality to bandwidth constraints. The paper presents different configurations depending on the type of game to be created. We also present a strategy for network gaming using heterogeneous devices focusing on the development of a game that allows users of mobile devices and desktop computers to interact and compete on a single domain.

Keywords: Mobile gaming, MMORPG, MIDP 2.0, Palm, Multi-Device Multi-player Gaming

1. Introduction

The gaming industry has seen a tremendous amount of growth in recent years. Possible reasons of such growth can be attributed to the various gaming platforms currently available on the market, such as gaming consoles, desktop computers, and mobile devices like the mobile phone and handheld computers. Another possible reason for such growth can also be attributed to the type of games that are currently being marketed. The top selling game(s) focus normally on role playing and competing with other players in the same game (e.g. Massively Multi-player Online Role Playing Games). Such games provide good user experience, challenging game play, and high levels of interactivity with other players on the same game session. However, given that most of these games target either desktop computers or home gaming consoles, the mobile arena is left largely untapped and unsupported. Major game developers are also focused on desktop gaming since these markets make up the majority of their revenues. Companies who develop for the mobile market are typically small to medium scale operations that focus and specialize on the mobile platform. In order to develop, support, and propagate gaming to the mobile platform, developmental and deployment issues would have to be identified and tackled. The goal of which is to eventually provide comparable gaming experiences on the mobile devices as compared to the desktop and in doing so tap into the growing mobile market. Aside from which, the experience should allow players to participate on the same game session regardless of whether they are using desktop computers or mobile devices.

2. The growing technological support on mobile devices.

A few years back, when mobile gaming would be mentioned, the first item that would come to mind would be the Nintendo® Gameboy™. This has since been reinvented into the Gameboy Advance™, and the Gameboy Advance SP™. Other mobile devices are also now being considered as a delivery medium for interactive gaming and entertainment contents. These devices would include other gaming devices, mobile phones, and the handheld computers. Recent improvements in the hardware resource itself allows for development of games on such devices that are now comparable to early gaming consoles such as the first Sony® PlayStation™. However, unlike the early gaming consoles, these new mobile devices have some advantages in that they now offer connectivity to the network whether wired or wireless, and they are portable.
Technological support can be classified into hardware support, and software platform support. Table 1 discusses the various hardware parameters involved that affect the type of games that can be developed for mobile devices, while table 2 discusses the software platform support on several mobile platforms.

Table 1. Hardware support for mobile devices.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Current Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>• Intel Xscale™ (400Mhz)</td>
</tr>
<tr>
<td></td>
<td>• Texas Instruments OMAP™</td>
</tr>
<tr>
<td></td>
<td>• Motorola MX™ Series</td>
</tr>
<tr>
<td></td>
<td>• Samsung ARM Compliant</td>
</tr>
<tr>
<td>Screen Technology</td>
<td>• Transflective TFT LCD with 64,000 colors</td>
</tr>
<tr>
<td>Screen Resolution</td>
<td>• 160x160, 240x320, 320x320, and 320x480.</td>
</tr>
<tr>
<td>Memory</td>
<td>• up to 128MB built-in and expandable up to 2GB</td>
</tr>
<tr>
<td>Graphics Accelerator</td>
<td>• ATI® Imageon™ graphics accelerator</td>
</tr>
<tr>
<td></td>
<td>• NVIDIA GoForce™ 2150</td>
</tr>
<tr>
<td>Connectivity Support</td>
<td>• Built-in IRDA, Bluetooth, and Wifi</td>
</tr>
</tbody>
</table>

Table 2. Software platform support for mobile devices.

<table>
<thead>
<tr>
<th>Spec.</th>
<th>J2ME</th>
<th>PalmOS</th>
<th>PocketPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Library</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2D Graphics Library</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Floating Point support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sound API</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Communication API</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IDE Support</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Size Limit</td>
<td>&lt;64KB</td>
<td>&gt;1 MB</td>
<td>&gt;1 MB</td>
</tr>
</tbody>
</table>

3. Graphics and Communication

Given the amount of improvement on the resources available to the mobile device, games are slowly improving in quality on such devices. Moving away from simplistic games such as the likes of Tetris™ or Chess™, and into richer games such as Tony Hawks Pro Skater 4™ and Doom II™ that are closer to what desktop devices are currently offering. However, a visit to [1] [2] [3] [4] would clearly show that aside from a handful of games, most the currently available titles are mostly if not all of them are single player or solo playing games. Following the evolution of computer gaming [5] it can be seen that mobile games are at the stage wherein the next innovation would be the development and deployment of multi-player mobile games in such devices.
Upon such a move, certain issues arises that would otherwise be irrelevant based on the older games previously developed. These are as follows:

- **Network Lag Time** – Unlike its desktop counterparts wherein games are typically played over a LAN or over broadband, mobile games would have to be played either through access points such as Bluetooth or Wifi, or via cellular technologies such as GPRS.
- **Unstable network connections** – compared again with its desktop counterparts, using cellular or wireless technology makes communication more susceptible to interference and disconnections.
- **Game play costs** – using readily available access points or cellular networks for mobile gaming connectivity also opens up the issue of cost in terms of how much will be spent per game session or game play. This would on the other hand raise issues on the frequency and length of communication that is taking place between the device and the server. Since telephone companies would charge based on usage and data transmissions, high traffic games would tend to get lower interests among players due to increased costs. However, for richer game play, this might sometimes be unavoidable. Thus, design from this point of view should be carefully considered.
- **Optimization of communication protocol** – given its inherent instability and the need for faster and smaller messaging blocks, studies and researches would have to pursued in order to derive certain design patterns or suggestion with regards to communications for specific types to games. The amount of data needed to the transmitted grows proportionally to the richness of the game user interface and user experience.

Typically, desktop games such as Warcraft™ and other MMORPGs do not consider cost in terms of amount of data transferred. However, these games do consider lag time in the sense that the interaction and interface halts to a stand still if no data was received during network lag instances. However, since desktop communication is more stable than mobile devices, lag time can be kept to a minimal. This however is not the case for mobile devices and hence the issue has to resolve so as to prevent undesirable game play experiences.

4. Game classifications and issues

Games can be classified in different categories in terms of game play, genre, and other measurements. For this paper, the concern and hence the classification is based on how games are being played and updated. Games are classified as single player games, multi-player games, turn-based games, real-time games, “Act Whenever” games, and slow update games. [6]

Discussions on the various requirements of the different types of games, and the discrepancies among mobile devices and desktop computers are stated in tables 3 and 4. With regards to table 4, this does not even take into account the several mobile devices among themselves already have at the least varying screen resolutions that posses a challenge to game developers. Although the issue is not very critical in the sense that the discrepancies among mobile devices are not high.

<table>
<thead>
<tr>
<th>Type</th>
<th>Cross Device Interaction</th>
<th>Game Graphics</th>
<th>Actual Communication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solo/Single Player</td>
<td>None</td>
<td>Low to High</td>
<td>None</td>
</tr>
<tr>
<td>Turn-based games</td>
<td>Minimal</td>
<td>Low to Medium</td>
<td>Minimal</td>
</tr>
<tr>
<td>Real-time games</td>
<td>High</td>
<td>Low to Medium</td>
<td>High</td>
</tr>
<tr>
<td>“Act Whenever” games</td>
<td>Low to High but not noticed</td>
<td>Low</td>
<td>Low to High but not noticed</td>
</tr>
<tr>
<td>Slow Update Games</td>
<td>Low to High</td>
<td>Low to High</td>
<td>Low to High</td>
</tr>
</tbody>
</table>
Table 4. Discrepancies between devices and desktops

<table>
<thead>
<tr>
<th>Area</th>
<th>Mobile</th>
<th>Desktops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen Resolution</td>
<td>Max. of 640x480 pixels</td>
<td>Typically 1024x768</td>
</tr>
<tr>
<td>Application size</td>
<td>1KByte(s) - 32MBytes</td>
<td>1MBytes - &gt;1000MBytes</td>
</tr>
<tr>
<td>Color and Sound Support</td>
<td>Black and White to 64K colors max. Sound limited to beeps and midi</td>
<td>64K/16bit colors to 32bit colors. Access to digital 5.1 channel devices</td>
</tr>
<tr>
<td>Power supply</td>
<td>Limited to battery life, typically 6 to 12 hours for game play</td>
<td>Unlimited, typically plugged-in to power source</td>
</tr>
<tr>
<td>Interruptibility</td>
<td>High due to possible incoming calls or other activities</td>
<td>Low due to dedicated resources</td>
</tr>
<tr>
<td>Networking Technology</td>
<td>IRDA, Bluetooth, Wifi</td>
<td>Modem, DSL, Cable, Ethernet, Wifi</td>
</tr>
</tbody>
</table>

However, regardless of the amount of discrepancies and issues, they would have to be resolved for the creation of such kinds of games. Resolution is not as straightforward as scaling graphics or performing tabular rule-based conversions of events and interactions. Here is where the type or classification of the games matter in the resolution process. A chess game can be easily translated across all platforms and devices, communication would also be simple due to its turn-based nature. However, games such as WarcraftTM would be a bit more difficult to translate due to the large discrepancies in screen resolutions. Possible workarounds could be lower screen quality and apply proper scaling, or allow for zooming of screens. Though these workarounds might give unwarranted advantages to those using better devices to play the game.

5. Playability Issues

The success of every game being developed regardless of its deployment platform has always been its design and the ability to properly market it to its desired audience. The physical difference and differences in the original purpose of the devices affect the way games are being designed.

For the desktop gaming industry, games should at least have 12 hours of playability and stay connected indefinitely until the game ends. [6] However, current games for mobile devices do not necessarily follow such accepted rules due to changing technologies and usage behavior. Thus, developing a game that will allow interaction between desktops and mobile devices poses greater issues. Basic contradictions among accepted gameplay behavior like long-term gameplay for the desktop while short gameplay for the mobile devices challenges even experienced developers in terms of how the game would be created.

6. Development Considerations

Based on our current work, we have attempted to implement a game that will exhibit some of the characteristics that we have been discussing. The following are some of the development considerations that we encountered.

A. Language

Early on we have to decide on the language we will use for the development of the game. The choices being C or Java. Considering the plan which is to develop two versions of the application running on two different platforms, and our own development experience we chose java.
B. Software Support

Our choice of language was partially influenced by the release of MIDP 2.0 of java. MIDP 2.0 provides the platform for better quality graphics with the use of sprites and easier networking development through the use of sockets. One of the issues we had to tackle about the use of MIDP 2.0 is the lack of devices that have already implemented MIDP 2.0 support in their mobile devices. The earliest phones that support MIDP 2.0 are slated to be released early 2004. In this light, we decided to implement the system using MIDP 1.0 to test some of the issues and just re-implement using MIDP 2.0 once the devices are available.

C. Game Choice

For the choice of the game that we are to develop, we looked at different categories of games and multi-player games as described in [6](Borella 2000). From these we identified the important communication and graphics requirements of each type as summarized in Table 3 as described in (Cheng 2004). We decided to create a variation of the classic battle city game. We made simple changes to the game play and changed it into a multi-player game. We decided to limit the graphics to 2D for faster processing, we wanted to ensure that any delays that are experienced by the game will mostly be due to network delays.

7. The Game : Tank Wars

Tank wars is a variation of the classic battle city game. The objective of the game is to either be the last one standing or to collect 10 gold coins. The game consists of the players (represented by tanks), the map (which is made up of bricks and cement walls), Power-ups (in the form of armor and bullets, gold coins). The game is played on a single map requiring at least 2 players.

A. Communication

In order to facilitate communication we decided to adopt the client/server architecture. A central server is created to host the game. All communication from each client is passed through the server and propagated to the other players. HTTP connections are initiated by each client device to the server whenever data is to be transmitted. The use of this architecture necessitated a look at 3 factors that may affect playability of the game namely bandwidth, latency, and host processing power (Kaukoranta and Hakonen, 2001). A dedicated web server was setup to act as the server program of the tank wars application. A java servlet was created to perform message routing and state consistency. Message that are being sent to and from the server have been minimized to minimize problems with large message sizes. A simple string message format was devised for passing and receiving messages. A diagram of the message format is show in Figure 1.

This represents every move made by one player,
If more than one move is made this trio is repeated for each move made by the player.

Figure 1 Message format

The use of messages was made as a decision to shift the burden of speed of the game away from network and to the device. This meant that the speed of the devices will also play a vital role in the speed of the game.
B. Game Consistency

The issue of data consistency is a very intricate matter. This directly affects the realism and responsiveness of the game. To maintain a consistent state of the game, we implemented a server based consistency mechanism in the game. The server keeps track of a move queue for each player. All the state of the game is kept by the server moves processed and displayed by the devices is based on the contents of the message queues returned by the server. The server is also responsible for resolving possible conflicts of moves or invalid moves made by a player. All players of the game use a two-way relay architecture for messages as described in (Kaukoranta and Hakonen, 2001).

A two-way relay architecture follows the following steps:

- A player sends his/her intended move to the server.
- The server checks if the move made by the player is valid, if the move is valid the move is updated.
- The server checks for collisions and resulting events from movements of non-player objects like bullets and coins.
- The player then receives the list of updated moves made by all players.
- The player then updates his/her system relative to the moves of all the players.

Since we are using a HTTP connection system instead of sockets, it becomes necessary to perform a fixed periodic polling mechanism to ensure the constant update of the game on each device. Every poll, the user sends its movement, and retrieves the updates for its screens and the screen is rendered again. During simulations we noticed that it is possible that there instances when there is already 5 moves made by one player while there is no move made by another, this poses a problem, the server is now unsure about the player who has not been making a move, is it simply a strategy of the player to lay in wait, maybe taking a defensive position etc. to solve this problem we decided to require all players to send a move request even if the move is the same as the current position. This process allows the server to evaluate if a player may have lost connection to the server already. A timeout value is assigned by the server based on the number of moves a player has missed. Once the limit is reached the player is automatically logged out from the game.

C. Latency Compensation

A latency compensation technique we tried to implement was the use of dead reckoning. Dead reckoning techniques are based on the navigational technique of approximating a players position based on a known position and direction (Kaukoranta and Hakonen 2001)(Borella 2000)(Griwodz 2002)(Gautier and Dior 1998)(Guo et.al. 2003). In the game we decided to check for possible moves made by a player if a player in position (x,y). If a move from a player is not received, the system will make an assumption of the possible movement to be made by the player and move one position towards the desired direction. We used this technique in a networked snake game that we initially created to test the server. It makes the client move one step forward when a move is missed even if the player has yet to make an update. We realized that using the same techniques on Tank wars was not practical.

D. Graphics

The most obvious problem that we encountered is the disparity between the different screen resolutions of each of the device we were using. The size of the PC monitor and the mobile device vary greatly. Our initial implementation made the screens independent, the only state consistency that we developed was the use of a grid to identify the position of each move. The first problem we encountered was the different implementations that were shown by different mobile devices of the mobile interface. This created an irregularity in the interface of the game in relation to the game. We solved the problem by requiring Every game instance to provide information about the device during registration for joining the game. The server then keeps track of the differing screen sizes of the different players and performs a mapping of the moves of each. Because the view screen of each player may not be the same, the server keeps track of the whole map and just keeps a boundary view of each player and their location relative to the map. Every players program keeps track of the whole map but only displays the screen that is best suited for the screen of the player. Although the view of the player is limited, Every move actually updates the whole map. This decision of creating views was a decision that forced us to sacrifice playability. It can happen that a player will suddenly see a bullet moving towards them without knowing who fired the shot.
8. Our Experiences

Aside from Tank Wars, we have developed several types of games that fit into various classifications and implemented in them different programming platforms. The games that were created include, a turn-based board game, two interactive arcade game, and a racing game. All of which are networked and can have opponents either on another mobile device or a desktop computer. For the mobile devices, it is assumed that the games are developed using a single platform and the issue device and platform is concentrated on the gameplay between desktop and mobile devices. Our games are limited to multi-session games, but only two players per game, hence multicast communication is not utilized as of the moment. Also, for the communication aspect, it is assumed that Internet connectivity would be provided regardless of medium used (e.g. Bluetooth, Wifi, or GPRS). The server modules were developed using either JavaServer Pages or Servlets. As of writing of the paper, the server is implemented differently and independently for each game. One of our future goal is to create a universal server for all types of games.

8.1. Turn-based board game

For this particular game classification, we implemented our own version of the Snake and Ladders board game on both the mobile device and the desktop. This is our baseline game in the sense that being turn-based, there is not a lot of issues on synchronization, communication delays and costs, and other game complexities. The only issue we had with this game is the varying screen sizes among various mobile devices and the desktop computer. In view of this, we resorted to developing multiple versions of the game for different mobile devices just to address the different screen sizes and graphics quality and capabilities. We decided not to perform scaling transformations for the games running on the mobile devices since the game is more of raster image based and not vectors and thus scaling would either result to quality lost or waste of storage space. As for the network communication, the only pertinent issue here would be the assurance that the turn token would not be lost and should be passed reliably. This task is delegated to the server to maintain stability, and on the mobile client side, polling is utilized since the server cannot notify the client device of changes due to lack of API support. For the desktop counterpart, polling was also used so as not to develop multiple versions of the server software of the game.

8.2. Interactive arcade game

In this type of game, we created two versions of games for this category. First, we created a game similar to the famous PACMAN and Battle City that supported two players, and mobile devices. The second game we created is similar to the arcade game Raptor.(Oliveira and Henderson 2003) Issues stated previously for turn-based games are also applicable at certain instances. However, for the communication aspect, different issues surfaced as compared to turn-based games. Synchronization, time-out, and network lag became more relevant in the sense that the moves and screen has to be updated in real time. Other issues also include the fact the for the mobile device, the game was developed using older Palm programming models which is not multi-threaded. Whereas, the desktop counterpart was developed using Java which now supports events and multi-threading. On the desktop version, if network lag occurs or there is a communication error, the game can proceed and measures would just have to be taken to handle the messages that will come in later. But for the Palm version, the program would tend to stop and wait for the message to arrive and responsiveness to time-out events are less effective. In our case, we decided to code the error handling specifically to the game itself since it is platform specific. For our next version, we decided to work with ARMlets for the PalmOS instead since it now supports threads in the application. This is an example where technological improvements can aid in providing solutions to existing developmental problems.

8.3. Racing Game

Finally, for the racing game, it is actually quite similar to the interactive game except for a minor twist. Unlike those top-view interactive arcade games, the racing game is a first person point-of-view type of game. This is advantageous to the game developers since they can now tweak the communication frequency and size specific to these types of games. There is no need to send and synchronize the entire game world since only one small part is visible at any one time. Like in the racing game, communication can be less frequent if no opponents are within your sight, since minimal delays or omissions in updating the track map would not be noticeable. This would help improve performance, scalability, and cost of the gameplay itself.
9. Conclusions and Future directions

This paper has shown that there has been great improvements on the capabilities of mobile devices that warrant attention to exploring the possibility of developing Massively Multi-player Ubiquitous Interactive Games. However, issues such as platform support, communication stability, costs, and proper game design should be resolved before wide-spread adaption would be possible. Discrepancies will never disappear since desktops, unless they become obsolete, will always be at least a step ahead of mobile device technologies. Hence formal workarounds should be looked into and defined for such a paradigm to become accepted and widespread.

It is the belief of this paper that the future of gaming on a ubiquitous environment, and applications development for such an environment in general would evolve around the following strategies:

- **Improvement of middleware support for game development** – the current set of support provided by middleware such as application servers deal mostly with client-server environments and business logic support. It is believed that in the near future, better support should and would be built into such middleware so as to support gaming itself and the use of mobile devices in gaming. More importantly, since such devices are portable, concepts of ubiquitous computing and ubiquitous gaming should also be supported directly by the middleware so as to ease the process of development and shorten the development period. (Hogie and Hutter 2003)

- **Hardware improvements** – it is projected that within a few years time, the processor, memory, and graphics capabilities of mobile devices would soon reach the standards to todays laptop computers. This would definitely fuel the development of mobile games that provide richer game play experiences, hence it would simplify consolidation of user game experience for the various devices in play. As of writing of this paper, screens that support resolutions of up to 640x480 pixels are already being developed and are soon to be made available to the market. [8]

- **Software Platform improvements** – it is projected that in the near future, software platform used for mobile application development would greatly improve to provide new features that is best suited for application and game development for the mobile environment. Possible improvements would include but not limited to location based routines, 3D routines, presence and instant messaging routines, personal information management (PIM) routines, security routines, and web services routines. [7]

- **Peer-to-Peer networks** – existing peer-to-peer applications deal mostly with messaging or file swapping. With the advent of technologies such as Bluetooth and better connectivity via GPRS or even 3G, peer-to-peer for gaming on mobile devices can now take no a different form. Users would in the future would be capable of creating adhoc networks that would allow for the simulation of trading activities within the game on the real world setting. The more traditional peer-to-peer networks over the Internet can also allow mobile devices and gamers to play against each other directly with greater scalability features than the current systems.

- **Improved communication infrastructure** – in the near future, the adaption of 3G or better cellular technology would allow for a decrease in the instability of the communication infrastructure so that games for the mobile devices would gain better performance and provide richer game experience.

- **Lowered recurring cost** – with the improvement of the overall technology and support, it is not improbable that the cost that will be passed on to consumers for using the infrastructure would go down. Much like the scenario of SMS technology, upon the widespread use of such technology, cost can be maintained at an acceptable level that still sustains profit for the providers as well as allow consumers to enjoy their benefits without paying a large amount that would otherwise deter the use of the technology.

- **Improved game design techniques** – studies are currently being performed to determine whether or not network lag really affects the game play. And if it does affect user acceptance of the game, up to what extent are players willing to accept and still be able to enjoy playing the game. This is extremely important for it not only may level the playing field of game play among various devices, it may also improve the performance and cost of gameplay since one can now optimize the messaging component of each game with these considerations in mind. (Vaghi et.al. 1999)

The paper also provided initial results in the development of a multi-player, multi-device application. Our goal of creating a game for multiple devices was realized our experience shows that the development of simple multi player games with “low to high” communication requirements and inter-device communication is pretty much the same as development of desktop applications. We feel that more work has to be done in moving the applications to devices that support MIDP 2.0. The use of MIDP 2.0 will open up the possibility of using UDP communication for different devices. The possibility of using this type of communication will allow the creation of first player shooting games
that are more interactive between multi-device applications. The increased graphics capabilities and processor speed will also make it possible to render better graphics for these games.

10. References

✓ April 2002 C. Grivodz, “State replication for multiplayer games”, Proceedings of the first workshop on Network and system support for games
✓ 1999, I. Vaghi, C.Greenhalgh, S. Benford “Coping with inconsistency due to network delays in collaborative virtual environments”, Virtual Reality Software and Technology Proceedings of the ACM symposium on Virtual reality software and technology London, United Kingdom Pages: 42 - 49 Year of Publication:

11. Web Site References


About the Authors

Danny C. Cheng is currently a faculty member of the College of Computer Studies De La Salle University Manila Philippines. He finished his master’s degree on year 2000 and has been teaching of seven years. He is focusing on the research field of net centric computing which includes web based technologies, mobile technologies, networking, and applicability of other fields such as artificial intelligence to it.
Email Address: chengd@ccs.dlsu.edu.ph
Mailing Address: College of Computer Studies De La Salle University 2401 Taft Ave. Manila Philippines 1004

Allan Christopher Chu is currently a faculty member of the College of Computer Studies De La Salle University Manila Philippines. He finished his master’s degree on year 2001 and has been teaching of six years. He is focusing on the research field of game development, user modeling, and similar or related fields of artificial intelligence.
Email Address: chua@ccs.dlsu.edu.ph
Mailing Address: College of Computer Studies De La Salle University 2401 Taft Ave. Manila Philippines 1004