This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE CCNC 2006 proceedings.

BlueGame – A Bluetooth Enabled Multi-Player and Multi-Platform Game: An Experience Report

Matthew M.H. Chan, Vincent Tam and King-Shan Lui

Department of Electrical and Electronic Engineering
The University of Hong Kong, Pokfulam, Hong Kong.
{vtam,kslui}@eee.hku.hk

***Abstract*— Computer games on mobile devices including cellular phones or handheld computers have become a fast expanding industry due to the recent advance in hardware and software supports. Most large mobile game and handheld vendors mainly focus on improving the interactivity and visual effects of, commonly single-user, mobile games. In this project, we carefully designed and then implemented an interactive multi-player action game, namely the BlueGame, transferrable between different computing platforms supported by the Bluetooth wireless technology. In addition to individual user's convenience to continue the multi-player game, our BlueGame prototype highlighted certain shortcomings of the existing Bluetooth technology, and more importantly our valuable experience gained for future wireless game development.**

I. INTRODUCTION

Mobile gaming on portable devices such as cellular phones or handheld computers has become a hot spot and fast expanding industry of developing attractive games on palm-sized consumer electronics due to the recent advance in hardware and software supports including fast processors, rising memory size and network bandwidth, sophisticated 3D display and rendering techniques requiring relatively less memory and bandwidth as compared to the console-level 3D graphical systems, and more importantly advanced mobile application development platform or toolkit to develop the increasingly complicated mobile games. An example of Imagination's new *PowerVR MBX Lite* [5] graphical toolkit providing a customized and powerful 3D engine as an integral part of the ARM's *Primexsys Wirless Platform* for mobile phones. The *PowerVR* 3D engine mainly uses tile-based rendering and deferred texturing to eliminate unneccessary graphics processing so as to minimize bandwidth and processing requirements in mobile networks and chips. Basically, tile-based rendering partitions a scene into small tiles, each of which can be rendered independently, whereas deferred texturing identifies surfaces in graphic elements that need not be visible in a scene when it is fully rendered to avoid redundant processing in conventional 3D system. Besides, Intel has developed its own 3D graphics toolkit customized for its XScale, ARM-based mobile processors. After all, it was estimated that mobile gaming revenues were $436.4 million in US alone, and $827 million in East Asia in the year of 2001. As predicted by the Datamonitor [5], wireless gaming will generate $17.5 billion in annual revenue worldwide by 2007. This clearly indicates the potential value, though with risks due to the new technological challenges and other environmental factors, of the mobile game industry in this decade.

For the current state of the art in mobile gaming, we observed that there were two major deficiencies. First, there was an unbalanced shift of efforts strongly biased toward the 3D graphical support so as to develop more attractive, mostly single-user, games on such limited screen and memory size of portable devices. Among the other mobile-game vendors, the two leading companies including Fathammer and THQ Wireless are working hard on this direction. Following the conventional design philosophy of computer games, we have no argument against developing fancy screens attracting more customers to play computer games. However, with the recent advance in wireless networks, especially the Bluetooth [1] or IEEE 802.11 (commonly known as the *WiFi*) technologies widely available in cellular phones or handheld devices [1], we can clearly see that there is high potential to *better utilize the available network bandwidth for* downloading the involved data more efficiently or even *providing more exciting and interactive multi-player mobile games*. Along this direction, Sega, a major game publisher, formed a new subsidiary wing named Sega Mobile in 2002 to develop mobile games for handheld devices. Only recently, game developers are starting to develop distributed, multi-player mobile games that are often server-based. An example is the Jamdat's *Gladiator* offering multi-player capabilities on mobile phones. Second, there are nowadays two major platforms including Sun MicroSystem's Java 2 Platform, Micro Edition (J2ME) [7] and Qualcomm's Binary Runtime Environment for Wireless (BREW) [5] for mobile game developement. Both platforms focus more on running complex games on *multiple types/models of devices* (e.g. different models of Nokia or Motorola phones) supporting the underlying platform. However, when considering each individual's need to continue playing a mobile, especially multi-player, game in different locations, it is clearly necessary for the mobile game to be

truly interoperable across different computing platforms, for instance from desktop PC to mobile phone, and vice versa, so that the relevant game settings and scores can be transferred from one platform to another to continue the mobile game. In view of the above consideration, we carefully considered and implemented a multi-player and multi-platform game, namely the Aeon Master BlueGame, based on Bluetooth enabled devices including mobile phones, and handheld, notebook or desktop PC. Here, we took an incremental approach to gather useful experience. Firstly, we had built a simple multi-player game called *BT Tontie* to test for the supports of the Bluetooth technology available for multi-player games. Then, we proceeded to build the targetted, *multi-player and multi-platform Aeon Master BlueGame* based on the previous experience gained. Up to our knowledge, the *BlueGame* system is one of the pioneering mobile games, if not the only one, in supporting both multi-player and multi-platform features through the seamless integration of both Bluetooth and Java technologies. Overall speaking, our *BlueGame* prototype has successfully overcome certain shortcomings of the existing Bluetooth technology, and more importantly helped us to gather some valuable experience possibly useful for developing other future wireless games.

This paper is organised as follows. Section II reviews the major supports of the Bluetooth wireless technology available for developing multi-player games, particularly our previous work on *BT Tontie*. In Section III, the overall software architecture, main difficulties encountered and valuable experience gained in building the multi-player and multi-platform *BlueGame* system are explained in detail. Lastly, Section IV concludes our work.

II. Our Preliminary Work

A. The Bluetooth Supports

This section briefly reviews several significant Bluetooth supports including the Bluetooth standard, Java 2 Platform, Micro Edition (J2ME) platform [7], Java Service Requests (JSR's) [7] and available software development tools like the BlueZ [2] and Impronto that we have used in *different stages of the BlueGame project*.

The Bluetooth [1] standard is originally devised as a wireless communication protocol targetted for peripheral devices of computers to reduce messy connecting wires. The protocol utilizes the radio frequency spectrum from 2.402 to $2.4835Ghz$, splitting into 79 channels, each with $1Mhz$ bandwidth, as an open band for scientific and hobby purpose world-wide. The same frequency spectrum is also shared by the IEEE 802.11 standard for the wireless local area network (WLAN). Unlike WLAN, Bluetooth devices are usually low-powered, and divided into 3 classes depending on their respective communication ranges. Any Bluetooth device initiating a connection is called the *master* whereas the device responding to the connection is the *slave* device. A master device can establish a *piconet* with 7 slave devices at most. Within a piconet, the frequency hopping sequence is totally controlled by the master device. A device can participate in more than one piconet to construct a *scatternet*. However, from our empirical observation, the network performance of scatternet often degrades drastically for many practical applications. Besides, the supports for scatternet is still unavailable in most mobile phones or handheld devices. Therefore, we would mainly focus on mobile games in piconets as far as our *BlueGame* project is concerned.

The software development platform we have chosen for this project is the J2ME [7] due to its wide interoperability. Besides the Symbian O.S. for mobile phones, many Palm^{TM} portable devices like the Tungsten 5 also support the J2ME. Basically, J2ME is a Java-enabled platform for portable devices with limited computational power and memory resources. Therefore, quite a number of capabilities including floating-point calculations in the Java 2 Standard Edition (J2SE) platform were withdrawn. Besides, to allow flexible extensions, the J2ME community can propose new profiles through the Java Service Request (JSR) in which each profile has certain requirement(s) on the underlying portable devices. An example is the Mobile Information Device Profile (MIDP) V2.0 requiring the device to have a screen size of at least 96×54 pixels and $128Kb$ of volatile memory for the Java runtime environment. After all, our *BlueGame* implementation heavily relies on 3 different profiles including the Connected Limited Device Configuration (JSR-30), MIDP (JSR-37, 118) and Java API for Bluetooth Wireless Technology (JABWT as JSR-82) [4]. For detail about the concerned JSR's, refer to [3].

Last but not the least, to provide the challenging multi-platform capability for our ultimate *BlueGame* system communicating between a Linux laptop and a Nokia 6600 mobile phone, we have used the BlueZ and Impronto packages. The BlueZ [2] is basically a Linux package to directly implement the Bluetooth standards for Linux. All in all, BlueZ offers the Berkeley Socket Interfaces [3] convenient for Linux programmers. Moreover, to use J2ME constructs and JABWT [4] in the Linux environment, we need the Impronto^{TM} [6] developer kit for Linux, freely available for any academic purpose. Intrinsically, the Impronto developer kit is still making use of the underlying BlueZ package to provide the JABWT interface for Java programmers to access any Bluetooth device. Since the later stage of *BlueGame* system involves the porting of a MIDP program to run in Linux, Impronto was used extensively at that period. Nevertheless, several bugs related to the combined uses of Impronto and BlueZ were uncovered during our porting. [3] contains a complete documentation of the uncovered bugs and their fixes.

B. The Multi-Player BT Tontie Game

After gaining sufficient experience in MIDP and JAWBT programming, we carefully designed and implemented our first Bluetooth multi-player game called the BT Tontie. The idea actually came from a popular online flash game called Tontie. The prototype implementation of our BT Tontie was a reduced version of the original Tontie game with poorer graphics and simplified game rules to at least support a multi-player community of more than 2 players with reasonable

interactivity, that involves timely updates to the latest game environment. Regarding this minimal requirement, our BT Tontie prototype was successfully tested on a Nokia 6600 mobile phone communicating with two other Nokia phone emulators running on two notebook PCs installed with the Bluetooth USB adaptors. Due to the lack of more mobile devices, we failed to prove the scalability of our BT Tontie prototype on more than 3 devices. However, from our empirical observation, we were confident that our multi-player game prototype should be scalable to all 7 devices within a piconet. After all, our game prototype of BT Tontie should be able to verify that the Bluetooth network is generally sufficient for *simple and less data-intensive* communications between devices in multi-player games.

Figure 1 shows the user interface of the BT Tontie game captured on the Nokia mobile phone. Multiple monsters could be probabilistically generated and then displayed at any of the 9 available cells. At the same time, any of the players can hit at any displayed monster to score with the instant updates performed on the whole multi-player community across the wireless network.

Fig. 1. Screen Capture of the BT Tontie

C. The Lessons We Learnt

About the Bluetooth connections used for our project so far, we had tested with the JABWT on mobile devices. It should be noted that such test could only be conducted on mobile devices or emulators readily implemented with JABWT, for instance the Nokia 6600 mobile phone. Since we had only one mobile phone, the actual tests on the Bluetooth connections were performed between a real Nokia phone and its emulator running on a notebook PC. Besides, we also conducted tests on the actual properties, especially the reliability and speed, of the Bluetooth connections using the RFCOMM against that of the L2CAP layer of the protocol stack for general applications. Initially, we chose to use the RFCOMM to establish the connections due to its implicit flow control. However, we found that there was high latency in the RFCOMM connections that were likely inappropriate for game applications demanding good interactivity. As a result, the L2CAP was consistently used throughout our game project.

III. The BlueGame

A. The Aeon Master BlueGame

The *Aeon Master BlueGame* is a multi-player and multi-platform mobile games that can be run on J2ME-supported and Bluetooth-enable mobile phones or Linux PC empowered with Bluetooth wireless technology. Due to the high portability of J2ME platform, given more time and resources, our BlueGame system can surely be ported to other handheld devices including the Palm^{TM} personal data assistants (PDAs).

In our multi-player *BlueGame* community, an aeon is actually a super entity that can be summoned upon to crush the enemy of the warrior. Within the game, there is a battle where two summoners summon Aeons for a duel. This is the original idea of this game. An aeon master has 0 to 5 aeons. He can discard an aeon and add new ones as long as the number is within the range. These aeons would be used in battles and gain experience through them. An aeon master created in one account, together with his aeons, can be transferred to a new host (either mobile phone or Linux PC) running the same game. All attributes would be transferred from one platform to another.

The design of our *Aeon Master BlueGame* is object-oriented. Figure 2 gives the overall class hierarchy of the multi-player and multi-platform BlueGame to show the main classes and their relationship. The class hierarchy includes

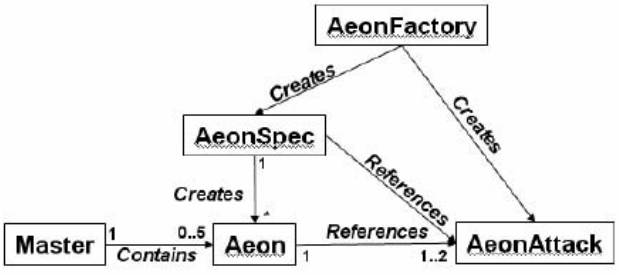

Fig. 2. The Overall Class Hierarchy of the Aeon Master BlueGame

an AeonSpec class which contains the description, thumbnail and all relevant static information about an Aeon. The Aeon, created from an AeonSpec, inherits all the details and contains some additional O.S. attributes. Clearly, it is not a good software practice to store all information of aeon within the Aeon class. Aeon descriptions and thumbnails are shared and static. Even if the player does not have a particular aeon, say Aeon #10, he/she can still access the relevant information. As a result, the specification and the instance are separated. Figure 3 shows the user interface of the Aeon Manager screen captured on the Nokia 6600 mobile phone.

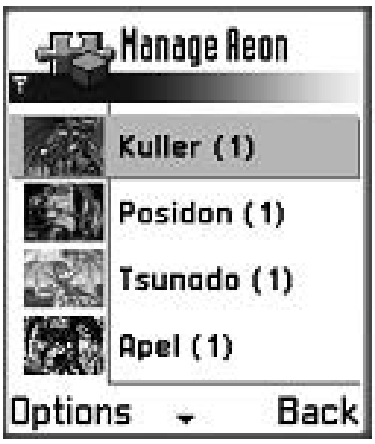

Fig. 3. The Screen Capture of the Aeon Master BlueGame

B. The Difficulties Encountered

- **StreamConnection flushing is not done instantly:** For general games, it is vital that data be sent to the other end on time and as soon as possible. In normal streams implementation, data are flushed out at certain timeout. We can call the *flush()* method and force the buffer flushed for a connection. However, the test reveals that data are not sent until it meets certain threshold. If the data to be sent is way below the threshold, there is no way to force the controller to send the data. As a result, StreamConnection cannot be used for real-time data transfer like multiplayer games. On the other hand, L2CAPConnection offers a reliable direct packet sending, and there is virtually no delay in sending and receiving. The flow control offered by the RFCOMM is not significant in game use, where data is usually real-time. All in all, we reverted to the L2CAP as the communication protocol.

- **Service search can be unstable:** An example of the unstable service search happened in the sending of data files from the the Nokia 6600 to the Nokia 6630 mobile phone. Although the Nokia 6630 appears in the searched device list, the sending was not successful. Later, we tried to connect from the emulator to Nokia 6630, but it always fails in the service searching part. But then when we switched our platform to Impronto, we found that both searching and connecting operations appeared to be normal.

- **The truncating problem of the L2CAP packets:** It was found that the size of L2CAP packets sent from our Nokia phone varies with maximum sent size. For instance, if it sent a 5-byte long packet before, all later packets would be truncated to no longer than 5 bytes. If we then send a 4-byte long packet, all later packets would be truncated to at most 4 bytes. To work around this truncating problem, we deliberately made the packets of a constant larger size.

IV. Concluding Remarks

In general, the limitations of Bluetooth technology for mobile gaming include the unstable speed fluctuation and severe signal strength fading, especially within crowded concrete buildings in some international cities like Hong Kong. These can be relieved by careful game design and powerful error correction mechanism. Moreover, in cases where flow control is not an essential element, it would be better to employ packet switched protocol such as the L2CAP instead of the stream-based RFCOMM, in order to obtain a better performance for mobile game applications.

Clearly, Bluetooth [1] and J2ME [7] are both new and evolving. It is not surprising to see that many bugs still existed with the new devices and implementations. Although it is feasible to develop multiplayer games over the Bluetooth wireless network, the current hardware and software supports could be discouraging. It is highly recommended that developers should try to investigate a more fundamental technology such as Symbian C++ [8] for its increased efficiency and control over hardware.

With limited time and resources, the tests we conducted within our *BlueGame* project could not cover every aspect of Bluetooth network gaming. A possible direction for future exploration should include the tests for advanced profiles in Bluetooth. With more stable release and possibly upgrades of the Bluetooth specifications, developers may find it more attractive to base their implementations on profiles such as PAN or OBEX [1]. After all, the potential of Bluetooth networked gaming is high and it would definitely be an irreversible trends for the years to come.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable feedbacks.

References

[1] The Bluetooth website at: http://www.bluetooth.org.
[2] The BlueZ website at: http://www.bluez.org.
[3] Matthew M.H. Chan. *BlueGame:A Networked Game on Bluetooth*, HKU-EEE Tech. Report, May 2005.
[4] The Java APIs for Bluetooth Wireless Technology website *available at:* http://www.jabwt.com.
[5] Neal Leavitt. *Will Wireless Gaming Be a Winner*, Computer Magazine, 2004.
[6] The Rococo Impronto Simulator Tool website *available at:* http://www.virtuosimo.com/rococoimpronto.html
[7] The SUN MicroSystem Java(TM) website *available at:* http://www.java.sun.com.
[8] The Symbian Developer C++ resources *available at:* http://www.symbian.com/developer/development/cppdev.html