Design, Development and Evaluation of a Chess Game in a Ubiquitous Environment

Vasileios Georgitzikis
University of Patras
8 PUBLICATIONS 80 CITATIONS

Design, Development and Evaluation of a Chess Game in a Ubiquitous Environment

Vasileios Georgitzikis[1], Christos Koninis[2], Ioannis Chatzigianakis[3]

{ichatz, koninis}@cti.gr, georgitzik@ceid.upatras.gr
1.Undergraduate Student, Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece
2.Graduate Student, Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece
3. Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece

Abstract

In the recent years we have seen an increased popularity in game development using Smartphones, which has provided an increasingly ubiquitous platform for designing games. In this paper we wish to investigate the use of a modern Smartphone's capabilities in game development by implementing and evaluating a classic game on the iPhone platform. We identify the limitations and possibilities that this field offers to the different aspect of game design.

Keywords: mobile devices, multiplayer games, pervasive games, mobile games

1. Introduction

Mobile phones have seen a huge growth over the last few years, and they are now starting to antagonize the sector of personal computers, with more and more people using their mobile phones to access the Internet while on the go, and to replace their personal computers and their laptops for most of their IT needs and activities. Therefore, it is no wonder that more and more IT companies and organizations are turning their eyes on these new platforms, one of which, arguably the most prominent one at the moment, is the iPhone.

One of the most important aspects of mobile phone development is gaming and most importantly, multiplayer games, due to the connectivity that the modern mobile devices offer. In fact, the need for multiplayer games on mobile phones is so great, that the iPhone is starting to take on major companies on the mobile console market, such as Sony with the PlayStation Portable, and Nintendo with the Nintendo DS, as we see on Figure 1 [1].

"iPhone (and iPod touch) is a gaming platform to be reckoned with. Controlling 5% revenue of a $10 billion industry in just a year and a half is significant. From a market share perspective, console games lost ground to portable platforms and iPhone. While the downturn in the economy may have dampened sales of the more expensive console games category, there is no denying that iPhone has generated substantial revenue and entered strongly into a mature industry." [1][1]

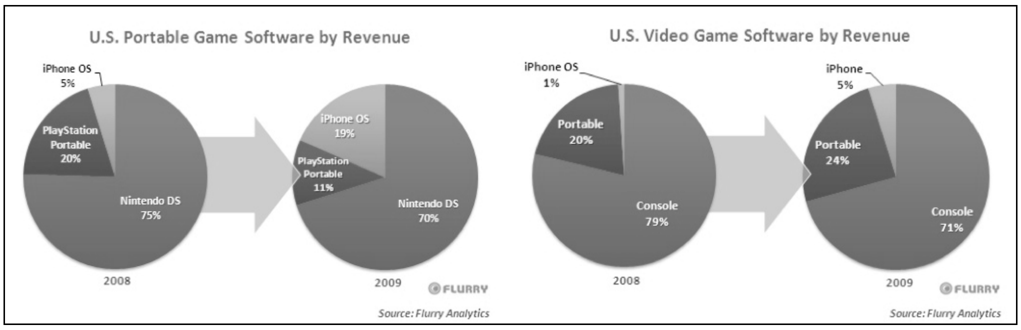

Figure 1. *Portable/Video Game Software by Revenue*

Mobile devices such as the iPhone can also be used for pervasive games, which is a huge academic and technological sector that is currently under heavy development. Modern mobile devices carry most, if not all, of the needed sensors for pervasive gaming, and since they are widely available, they can be used to spearhead pervasive gaming development.

In order to accurately depict the current situation on mobile game development, we asked ourselves what makes a good game. Certainly, we had to take 3D into effect, since it is a prerequisite in almost all computer games. Given that the iPhone is acclaimed for its hardware-accelerated 3D performance and capabilities, we wanted to test how easy it is to develop 3D games for the iPhone, and measure that performance. The second thing that came to mind was the multiplayer aspect of gaming, so we decided to implement device-to-device interconnectivity via Bluetooth. And last but not least, we needed something everyone would recognize and know how to interact with. Therefore, we chose to create a chess application with 3D pieces, and multiplayer connectivity, because it would allow us to test the capabilities of the device when it comes to complex 3D rendering, and it is a great multiplayer game. It would also give us the ability to experiment with Human Computer Interaction, and study its effects on the gameplay of a well-known game.

When designing a good mobile game, we also have to take into account the capabilities of the platform and act accordingly. First of all, how complex 3D graphics can we display before the game becomes unresponsive? And if it is responsive, how much battery does it consume while our program runs? And last but not least, the network limitations have to be taken into consideration as well, since mobile devices have a limited bandwidth, unsuitable for, say, a First Person Shooter.

2. Previous & Related Work

There are numerous chess applications already on the iPhone, but most of them underutilize the available hardware. There are a few multiplayer applications using two or more devices to interconnect with each other, or with another Chess application through the internet, such as Chess with Friends and Chess Online Expert [2] but do not support real 3D chess pieces[3]. In regards to other devices, the situation is even worse. This is mostly due to the fact that other devices lack the processing power of the iPhone, and, of course, the market is much smaller and there is a lack of incentive to develop applications for these platforms.

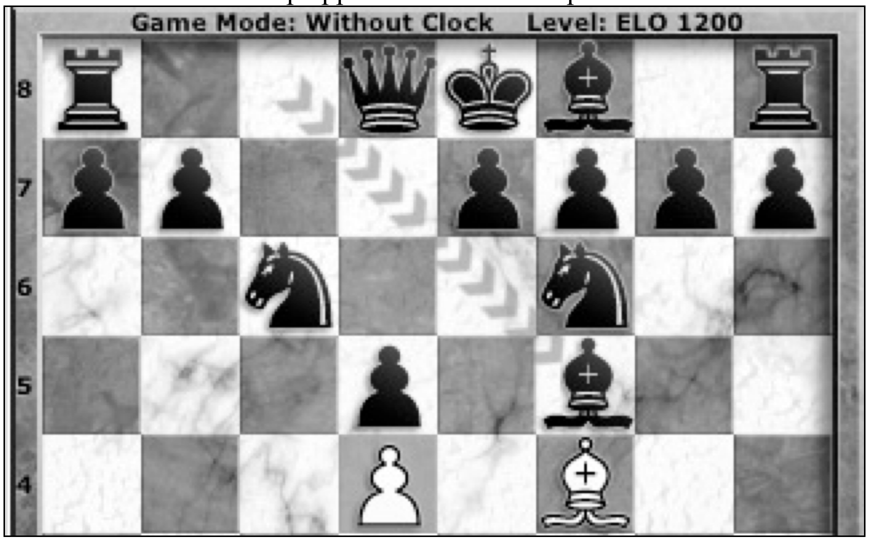

Figure 2. *Fritz Chess, a Chess Application for the iPhone [3]*

3. Underlying Technologies

From a Hardware perspective, the iPhone has all the hardware you would expect to find in a modern mobile phone, including a Camera, a GPS, GPRS/3G connectivity, Bluetooth and WiFi connectivity, a Speaker (along with a Microphone and Headphone), an accelerometer, a multi-touch screen, and last but not least, a light and a proximity sensor.

It also has a connector via which you can add additional input/output devices and sensors. Therefore, it becomes obvious that the modern mobile phones are a great tool for developing applications with rich HCI. For example, it provides gesture recognition which we can use to let our user control our application.

The iPhone was a revolutionary platform when it was introduced mainly for one reason, and that is its HCI abilities [4]. Being the first mobile device with true multitouch, accelerometers, and really intuitive interface, it had a huge impact on the way people interact with their mobile phones. This also gives developers a lot of tools for creating applications with good HCI.

"If there's anything revolutionary, as Apple claims, about the iPhone, it's the user interface that would be nominated. Countless phones make calls, play movies and music, have maps, web browsers, etc., but almost none seem able to fully blend the experience--which is part of the reason people flipped out at the idea of an iPhone" [4]

On the software side, the iPhone uses programming languages based on C, particularly plain C, and an Object-Oriented superset of C, Objective-C, to access the built-in Frameworks, as well as HTML, CSS, and Javascript for writing web applications that can then be added integrated into the iPhone with the look and feel of native iPhone applications. For 3D Graphics development, the iPhone uses OpenGL ES, a light version of OpenGL designed primary for use in embedded and portable systems. OpenGL ES give developers the power and advantage of a robust graphics framework, based on the very popular OpenGL framework which is used by a lot of the biggest game development companies for PCs[5], which also has the advantage of letting developers dive in to the iPhone Game Development very easy, with only a small learning curve required.

The iPhone also uses established internet technologies such as sockets to allow for multiplayer gaming (and not only), using its built-in Bluetooth and WiFi interconnection. The iPhone also uses a Model-View-Controller architecture, which isolates the application logic from the presentation, and user input.

Compared to the competition, the iPhone shines for its tools, ease of use, and market share. In regards, to the Android platform for example, Android just implemented native OpenGL using C instead of Java, which used to have a huge toll on performance, and it still lacks the market share and mindset to take on the iPhone as a platform. However, it has the largest growth rate, so it is a promising platform [7].

In regards to the Windows Mobile platform, until recently it had fallen behind, both in market share, features, and of course Mobile Gaming, which never bloomed on the platform. However, with the introduction of Windows Mobile 7, Microsoft has introduced the necessary development tools to develop games for Windows Mobile using XNA, the same technology used for the Xbox 360, and Silverlight. Even at the unveiling of Windows Mobile 7, the demos were astonishing [8].

4. Architecture

The design of our architecture takes advantage of all the modern concepts and software engineering techniques that ensure easier portability and more stability. On that grounds, we use the Model View Architecture, which we will discuss below. Our architecture is comprised of a number of individual components that you can see in Figure 3. On top of everything sits the User Interface layer, which is responsible for showing the output of our program to the user, and getting the input. In the middle, we have the user/network input layer. These layers, as the name suggests, are responsible for taking input from the user(via the user interface), or from the connected device, and passing it to the game engine. Finally, on the bottom we have the game engine of the application.

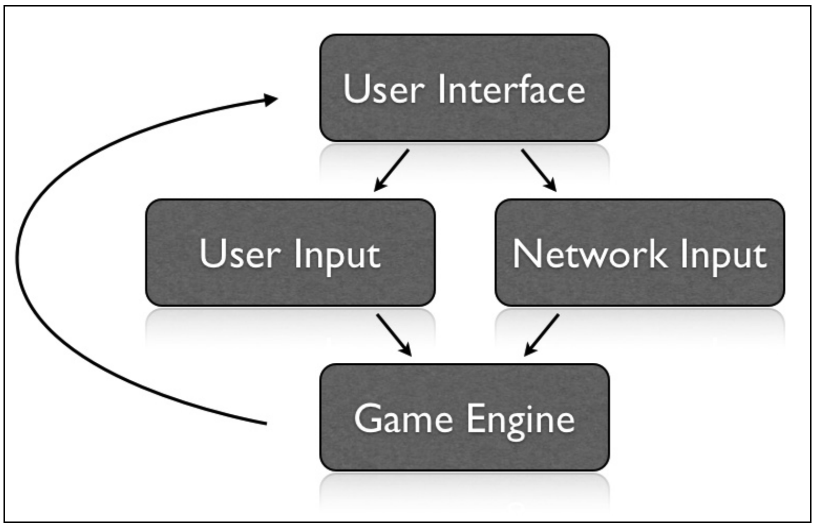

Figure 3. *Underlying Architecture*

Engine

The engine component includes two vital modules. The game logic that is responsible for performing movement validity checks, specialised movements such as Castling, and generaly ensuring the compliance with the game rules. The second module is the graphics module, which is responsible for rendering the models and the 3d space transformations that take place with movement.

In the game logic module, we represent the game information using different matrices for storing pieces position, players' pieces, and model arrays. More specifically, we have a 8x8 matrix that stores the player who has a piece on each square, the kind of chess piece on the square, and a pointer to the model array. The model array is a set of vertices that describe our models on the 3D space. We also have a 8x8 matrix that holds the valid positions when we chose to move a piece. This is used for highlighting the valid movements on the chessboard when a user chooses a piece, and for checking if the selected move is legal. This module also holds the previous state of the game, in order to provide an undo function for the user.

The graphic module is responsible for rendering the models. The format used for the models is a simple array of vertices that describe them in the 3D space. The pieces are rendered inside the main loop of the game. Since we do not need to render pieces that have been captured and are no longer in the game, we keep track of the pieces to let the graphic module know whether to redraw the piece or not. This is information is changed by the game module when needed, and thus we do not render unneeded pieces.

Model View Controller Architecture

By taking advantage of the iPhone MVC architecture,we created a universal binary for both the iPhone, and the new Tablet PC by Apple, the iPad and easily targeted a broader audience by developing an application that will run natively on different platforms. Using the MVC architecture, we created a new View element to fit the iPad's dimensions, and to be able to make use of the iPad's additional APIs if needed. Since Apple's Darwin Kernel that the iPhone uses allows an executable to have multiple types of execution code inside, for example, for multiple architectures, we can create a Universal application, that will work both for the iPad and the iPhone, using the appropriate View for each device.

In addition to using the MVC architecture, we needed to isolate the inter-device communication from the rest of the program. If we had a single-player game, the flowchart would be this:

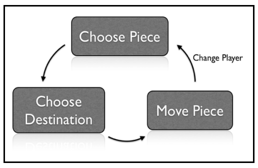

***Figure 4**. Single Player Architecture*

Networking

In order to enable multiplayer capabilities in our games, we used Bluetooth to connect two iPhones running our chess application, to allow two users using an iPhone each to play a chess game with each other. This can also be achieved using the WiFi connection of the iPhone, which can be used for a lot of interesting uses such as connecting an iPhone with another chess application, connecting the iPhone to a remote application acting as a chess engine, to provide a means to play against the computer using an AI, etc. In fact, in the last case, it is even possible to create a server that will save the state of a match mid-game, then load it when the user connects again, and provide statistics such as the percentage of wins vs loses, and even adjust the difficulty of the AI based on the user's win/loss ratio.

In regards to communicating between chess applications, we use the most common protocol, the Internet Chess Server protocol, which is also used by the leaders in web chess, XBoard, and its Windows counterpart, WinBoard. [6] This gives us the ability to interconnect with other popular applications running on different hardware, and the ability to connect with Internet Chess Servers to play games.

With two intercommunicating devices, we now have the following flowchart:

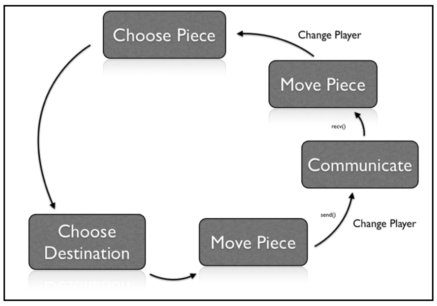

Figure 4. *Multi Player Architecture*

Because the iPhone uses sockets over bluetooth, and Apple's implementation of the Zeroconf service discovery protocol, Bonjour, it's easy for a programer to implement bluetooth communication and discovery. We used bluetooth to let two users, using an iPhone each, to play a multiplayer chess game via bluetooth. Each user controls one set of pieces (black or white), and each movement is passed on to the other device via bluetooth.

User Interface

The main component of our UI, the board, uses tapping to move the pieces around. When the user taps on a chess piece, we highlight the available movements, and when the user taps on the destination, we move the piece, as seen on Figure 5. We also have 4 buttons that allow the user to rotate the board and reset the rotation, undo the last movement, and reset the board to its original state.

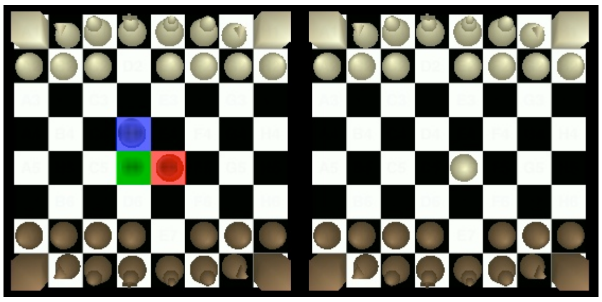

***Figure 5**. Graphical User Interface*

As we mentioned above, the iPhone is a great tool on which to build rich HCI applications. For example, we make use of the accelerometer of the iPhone to implement “Undo Movement” by shaking the device, or rotate the chessboard in the 3D space by rotating the iPhone itself. This makes our program much more intuitive, and it also saves a lot on screen real-estate, which can be used to show more information, which is a huge plus, given that one of the most prominent disadvantages of mobile devices in comparison personal computers is the small screen of those devices.

5.Evaluation

We wanted to test the iPhone’s power when it comes to OpenGL 3D graphics. To do that, we created the same application with a varying number of polygons. The application was a chessboard, which is a textured square consisting of 2 triangles, and the chess pieces, which consisted of a different number of polygons each time, and the user can turn the rotation of the chessboard on and off. We measure the Frames Per Second and the response time of the iPhone when we press a button, in relation to the total number of polygons in the application.

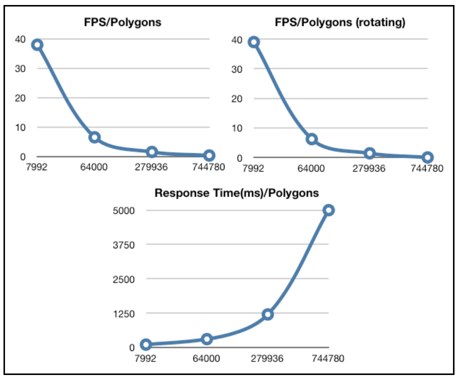

***Figure 6**. Frames Per Second and Response Time*

It is easy to realize that, in order for the game to be playable, and responsive, the limit is around 64.000 polygons. We then tested the 3D graphics acceleration power of the iPhone on a real game. We recorded the frames per second on the final chess board, which has an overhead due to the checks for changes in position, and the overhead of calculating the new positions for the chess pieces (for example, when a chess piece moves), and calculating the changes when a piece destroys another piece, etc. This recording is meant to provide results for performance in real-world scenarios.

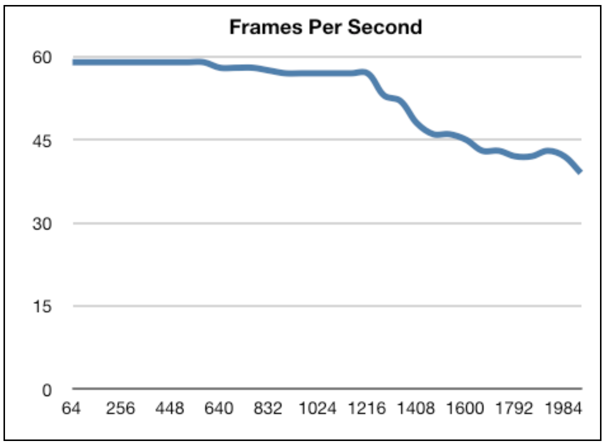

***Figure** 7. Frames Per Second in the Real World*

As we can see, our application is perfectly playable, while at the same time providing good 3d graphics. There is also a threshhold at 60 frames per second, which is why we see that there is no difference when the number of polygons drops below 640. It is very important to note that the user cannot tell a difference when the fps is over 15, so there is no visual difference between using 1984 or 64 polygons.

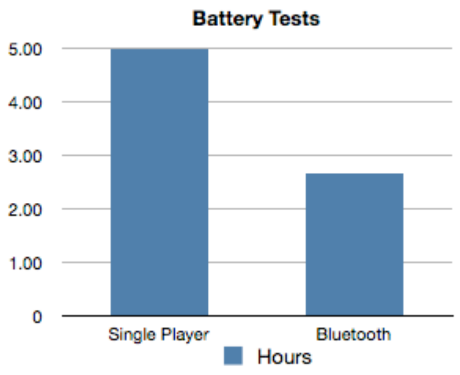

***Figure 8**. Battery Time*

We also measured battery consumption while playing the game with bluetooth enabled and disabled. With bluetooth disabled, the battery level dropped 10% in 30 minutes, which, by extrapolating, means that our application can run on a fully-charged iPhone for about 5 hours. With bluetooth enabled, it took 16 minutes for the battery to drop 10%, which gives us 2.5 hours of playtime, nearly half that without bluetooth. However, it is still a lot of playtime, and certainly within acceptable range for a chess game.

8.Conclusions and Future Work

In conclusion, we believe that the iPhone is a very promising platform, and mobile gaming in general will be a very important academic and commercial field. We would like to make use of more of the built-in technologies, such as the accelerometer to rotate the board, and we would like to try techniques to reduce the battery consumption of our program. For example, we would like to try setting the maximum FPS to 20, since it does not make a difference to the user, and see what difference it makes on battery consumption.

But the most important networking aspect is the ability to connect to the Internet from the device. This means that we can create applications that can connect to a server that keeps statistics, or applications that let you play multiplayer games with hundreds of other players on the Internet, regardless of location and country. Our goal here was to follow the most commonly used chess protocol and connect with existing Internet Chess Servers. We can even possible for a game to dynamically download 3D models from the internet when loading, so we, as the programmer, could just change the file containing the model in the server, and all players will therefore have the updated model, with no application updates required.

9.References

1.http://blog.flurry.com/bid/31566/Apple-iPhone-and-iPod-touch-Capture-U-S-Video-Game-Market-Share

2.http://www.pocketgamer.co.uk/r/iPhone/iPhone+news/feature.asp?c=10054

3.http://itunes.apple.com/app/fritz-chess/id310167298?mt=8

4.http://www.engadget.com/2007/07/03/iphone-review-part-1-hardware-interface-keyboard/

5.http://en.wikipedia.org/wiki/OpenGL

6.http://tim-mann.org/xboard.html

7.http://www.electronista.com/articles/09/05/11/android.to.grow.900pc/

8.http://www.engadget.com/2010/02/15/windows-phone-7-series-hands-on-and-impressions/