EECE 496

Analysis and Modification of

THE FUGITIVE

LOCATION-BASED GAME FOR WIFI DEVICES

Name:	Si Colleen Qin
Student Number:	84698018
Supervisor:	Sidney Fels, Rodger Lea
Project Code:	SSF-W06-01
Project Partner:	William Tsui

Submission Date:	April 12, 2006
Version:	1.0

ABSTRACT

The Fugitive, formerly known as Chase Bob, is a location-based Wifi game played on the UBC campus. The objective of this game is for a team of three players to seek, track, and surround a virtual person, Bob. In order to communicate with the other team members and to determine the position of all parties, each player uses a tablet PC that is connected to the UBC Wifi network.

When the 496 project team was enlisted by the UBC Ubicomp group to complete game implementation, the software of The Fugitive was at the end of a development iteration phase. Most of the client (user) components were finished for playability; however, the game had not been tested on the UBC campus. Therefore, positioning accuracy, Wifi connectivity, and usability were all indeterminate. In addition, the server replay tool that was vital for future research analysis had very limited functionality, and significant feature additions were necessary.

Following field testing, a number of tasks were done to improve the game. To increase positioning accuracy, the project team war-drove throughout the UBC campus and mapped GPS locations to Wifi access points. The connectivity of the Wifi network was characterized, and a new playing field for the game was determined. Software modifications for game usability, such as game parameters and UI features, were also implemented. The replay tool was changed to include ink widget display, stroke display, scroll bar of ink history, and Bob's location. Throughout the project, a primary challenge was Wifi connectivity and UBC wireless login for the game. This was solved with the addition of a software component to log into the UBC wireless network automatically whenever connection is lost. These modifications improved the usability of the game, and the 496 team effectively met the objectives of the project as determined by the Ubicomp group. The resulting state of The Fugitive game is playable and analyzable based on recent team testing.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

GLOSSARY

GPS	Global Positioning System: A worldwide radio-navigation system that is widely used in marine, terrestrial navigation and location based services.
PEAP	A protocol developed jointly by Microsoft, RSA Security and Cisco for transmitting authentication data, including passwords, over 802.11 wireless networks. It authenticates wireless LAN clients using only server-side digital certificates by creating an encrypted SSL/TLS tunnel between the client and the authentication server. The tunnel then protects the subsequent user authentication exchange.
WEP	A security protocol for Wifi networks defined in the 802.11b standard. WEP is designed to provide the same level of security as that of a wired LAN.
Wifi	Refers to any type of 802.11 network (802.11b, 802.11a, dual-band, etc.)
WPA	A Wifi standard that was designed to improve upon the security features of WEP. There are two main improvements when compared to WEP: • Improved data • User authentication

LIST OF ABBREVIATIONS

- CWL Campus Wide Login
- GPS Global Positioning System
- PEAP Protected Extensible Authentication Protocol
- UBC University of British Columbia
- Ubicomp Ubiquitous Computing
- WEP Wired Equivalency Privacy
- Wifi Wireless fidelity
- WPA Wi-Fi Protected Access

1. INTRODUCTION

The Fugitive, formerly known as Chase Bob, is a mobile location-aware game based on the UBC Wifi network. In this game, a team of 3 players attempts to capture a moving virtual person, Bob, by surrounding and tracking him. The game is a collaborative project between the HCT (Human Communication Technologies) lab and the UBC Ubicomp group. This 496 project was enlisted to test and to analyze the game's functionality, and to make appropriate modifications for improving game usability.

A number of objectives were determined for the project. Firstly, it was important to field test the current application to validate whether established user requirements have been met and to identify possible improvements. It was determined after a number of these tests that characterizing the UBC Wifi network was necessary. The positioning accuracy achieved through the Place Lab software tool also needed to be improved. In addition, connectivity to the UBC wireless network was poor, and a solution was to be found. The server replay tool for game analysis lacked a number of features, and this tool also needed to be further developed. These objectives all contributed to the goal of implementing a functional and playable game by iteratively making it better.

This project's importance relates to the Ubicomp group and the HCT lab's hopes of using The Fugitive game to examine social navigation, collaboration strategies, as well as team communication based under a location-aware environment. Upon completion of the game, Ubicomp and HCT will be able to collect data from game plays that will hopefully advance research in ubiquitous computing and location awareness fields.

There are a number of stakeholders for this project. They include the following EECE and Computer Science professors, graduate students, and undergraduate 496 students:

- Dr. Sidney Fels: HCT lab director and the EECE 496 main supervisor
- Dr. Rodger Lea: Ubicomp group lead, The Fugitive group lead
- Mike Blackstock: Software designer
- Phillip Jeffrey: Usability and test lead
- Meghan Deutscher and Tony Tang: Game usability and testing engineers
- Colleen Qin: EECE 496 student – usability testing and software improvement
- William Tsui: EECE 496 project partner – usability testing and software improvement

Mike Blackstock and Phillip Jeffrey were the main interfaces for this project. Mike had been the primary software developer for the game prior to the involvement of the 496 team, and he had implemented the game to a testable state. Phillip, the usability and test lead, was the person who best understood all functional requirements and standards for the game. These two members were both actively involved throughout the project and provided valuable feedback for possible game improvements.

The 496 project partner for The Fugitive development was William Tsui. Due to the group-orientated nature of this software project, a number of tasks were done with William. These tasks included usability testing, feature validation, Wifi characterization, and location accuracy improvement. In addition to these, there are several tasks that were also done independently with feedback from the other team member. While William focused on the replay tool and game parameter modification, the independent part of this particular 496 project included composition of the game's user manual and the development of an automatic UBC wireless login software component for improved connectivity.

In the following sections of the report, all of the above topics and tasks will be presented to provide appropriate context and information. However, individual and team tasks that were done pertaining to this project will be discussed in detail while independent components implemented by William will only be briefly mentioned. Furthermore, since this report focuses on field

testing and implementation improvements of The Fugitive game done by the 496 project, prior software development will not be discussed.

The following main topics will be presented in the upcoming sections:

- Initial field tests and findings
- Wireless positioning accuracy
- UBC wireless connectivity
- Replay tool
- Client usability
- Testing and usability results discussion

2. INITIAL FIELD TESTS

At the beginning of the 496 project, the game was at the end of an iterative development phase, and it was important to determine the current usability as well as the technical state of the game via field testing. By doing so, immediate objectives and future game improvement tasks can then be established. A number of these tests were carried out throughout the first month of the project, where multiple players tried playing the game. The following findings on positioning accuracy, connectivity, and usability were obtained from the tests.

2.1. POSITIONING

The Fugitive uses a map of UBC as its game background and displays all 3 players at their detected location. The positioning accuracy as measured from the field tests was poor. The difference between the detected location and the actual location of players usually ranged between 40 – 50 meters. For some locations, the position inaccuracy was as significant as 100 meters. There was also a skipping behavior exhibited by players on the map. Frequently, with little or no actual movement from a player, his/her corresponding location on the map would skip from one location to another location that's more than 100 meters away, then quickly skip back within a matter of seconds. Sometimes, the player would skip to many locations on the map that were within a 100 – 150 meter radius. This constant skipping of players was problematic as location accuracy was vital to the playability of the game.

2.2. CONNECTIVITY

The client (user) component of game, which runs on the tablet PC carried by each player, needs to be constantly connected to the UBC wireless network. This allows each player's location and messages to be shared amongst the other players in the game. To do this, the clients synchronize with the game server wirelessly, then the server broadcasts to all players any updated information. During the field tests, the game was frequently disconnected, which prompted players to manually sign in to the UBC wireless network numerous times through UBC's

wireless login website. This was very troublesome, as the players were constantly interrupted in the game for login. In addition, typical game players are expected to be non-technically adept, hence in a real game, many of them would not understand why the game is not updating new information, nor would they know how to resolve this connectivity issue. This is a major challenge for the game playability.

2.3. USABILITY

A number of usability issues were found when playing the game. Firstly, the UBC map used for the game was larger than a tablet display screen, but it was not scrollable. Hence, users would only be able to view the entire playing area on the map if the tablet PC was in portrait mode. This was impractical for players who wanted to change the tablet display orientation for personal preference or for accessing the keyboard. Secondly, the game parameters that navigated Bob proved to be too difficult for players to capture him. When players tracked and chased Bob, Bob would move away very fast such that it was nearly impossible to catch up to him, much less surround him and win the game. Also, the playing area of the game included both the north side and the south side of campus, ranging from the Buchanan Buildings on the north, to the Forestry Building on the south, and spanning between Westbrook Mall and SW Marine. This large playing area affected the time to play the game. The allotted time for a complete game had been determined to be 30 minutes, as holding a tablet PC continuously can cause severe arm-fatigue beyond a half-hour threshold. This 30-minute time span included both of the 2 phases of the game: finding Bob, and capturing Bob. However, during field tests, it took more than 30 minutes to complete the first phase of the game. The second phase of the game was attempted for more than 45 minutes, but was determined by all players to be impossible to finish in reasonable time. These issues were all important usability challenges that affected the playability of the game. They would need to be addressed.

3. POSITIONING ACCURACY

From the results of the field tests, it was determined that positioning accuracy of the players in the game was poor and would need to be significantly improved. To do this, further analysis was necessary to find the cause of the inaccuracy, and an appropriate solution can then be found.

3.1. POSITIONING ACCURACY ANALYSIS

To investigate the positioning accuracy problem further, war-walking was done throughout the UBC campus. This involved trekking in various areas and logging how much location accuracy of the game varied. It was found that in most areas on the south side of campus, near MacLeod and ICICS buildings, positioning accuracy was within 20 to 30 meters. The position accuracy along University Boulevard was also acceptable at around 20 to 40 meters. However, on the north side of the campus, such as near the Koerner library, Math Building, and Agricultural Road, skipping behavior of players were exhibited in the game, and positioning accuracy was around 50 to 100 meters. This collective data showed that while position accuracy was very good in some locations, it was dismal in others.

The most probable explanation for this behavior was that only a limited number of UBC Wifi points were used to determine the position of the players, as only certain Wifi points were in the Place Lab database library used by The Fugitive game. This is discussed in detail in the following section.

3.2. PLACE LAB

Place Lab is a software library used by The Fugitive that determines positioning by listening to radio signal beacons, such as 802.11 access points [1]. By using a tool called MapLoaderGUI, Place Lab maps detected access points to the access points in the database, and looks up the corresponding GPS coordinates for 3 access point with the strongest signal strength. If any of the access point is not in the database, Place Lab then finds the access point with the next strong

signal. This process continues until 3 access points with GPS coordinates are found. The Fugitive software then uses these coordinates to calculate the location of the player through a triangulation method. This is how the locations of players are obtained.

Poor positioning accuracy in some areas could be attributed to the lack of detectable Wifi access points at that location with GPS coordinates in the MapLoaderGUI database. It is likely that the game cannot map nearby Wifi access points with GPS coordinates to calculate location; therefore, it is forced to use access points further away. This increases the error of the calculated result, and produces a location that is inaccurate. In addition, multiple Wifi access points from distant locations could have similar signal strength. Hence, the game would try to use one access point to determine the location, and then quickly use another one because the latter one's signal is now slightly stronger. In typical scenarios where the access points are close by, the result wouldn't differ much. However, with access points far apart and from different directions, the location obtained could vary drastically, thereby producing the skipping behavior that was observed during field testing.

3.3. WAR-DRIVING

To validate whether the above hypothesis is right and to improve positioning accuracy, war-driving was performed. This involved walking and driving throughout the UBC campus while using a software tool called Network Stumbler to detect nearby access points. A GPS location device was connected to the software tool such that each detected Wifi access point was mapped to a GPS location. This was done throughout UBC, as seen by the blue lines in figure 1 on the following page. The resultant collection of Wifi access point data was then uploaded to a www.wigle.net database. This database allowed Place Lab's MapLoaderGUI component to synchronize and obtain newly updated data.

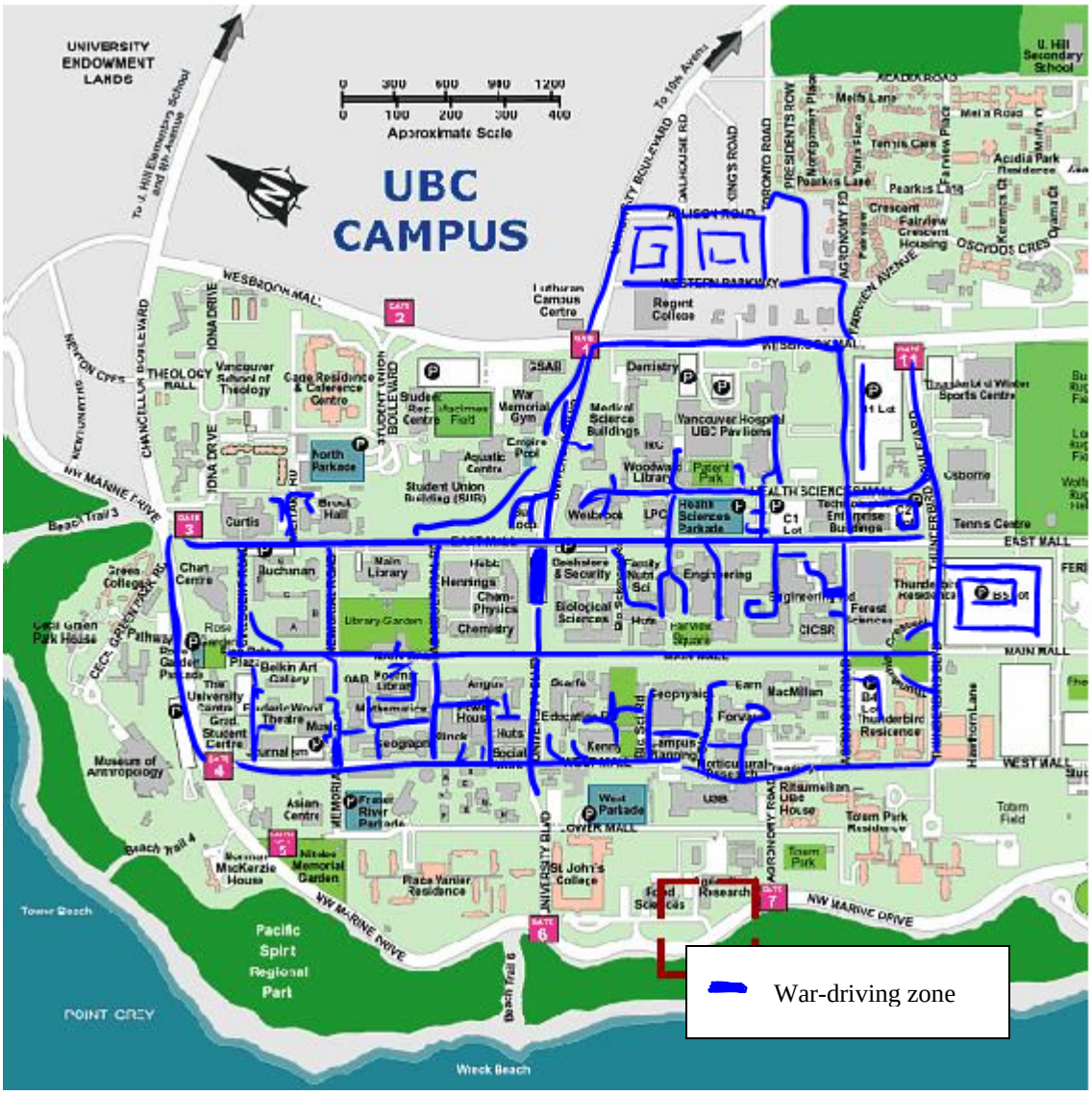


Figure 1. UBC War-Driving Areas

War-driving increased the number of access points pertaining to the UBC campus area to include around 1000 new beacons, increasing the total number of access points at UBC from 2400 to 3400. The result of adding these new access points was very positive. Positioning accuracy for The Fugitive now improved to between 15 to 20 meters of error for most areas, and around 30 to 50 meters in locations of low Wifi signal strength. These changes enabled the game's estimated location for players to be accurate enough for game playability.

4. UBC WIRELESS CONNECTIVITY

Despite positive results achieved with positioning accuracy, issues relating to UBC wireless connectivity posed a significant problem. Poor connectivity would prevent The Fugitive from being a functional and usable in a large-scale. Frequent disconnections from the UBC network interrupted the game and required the user to manually connect back through the UBC login website. This manual login procedure is a primary pain point from the user's perspective. This is because many times, the game would only be disconnected momentarily, yet another login is required to continue playing t he game. In addition, the user frequently wouldn't even notice that connection was lost. Although the connectivity status is displayed in the game, it is not very noticeable, and thus players may try to continue playing the game. This can prove to be confusing and frustrating.

4.1. UBC WIRELESS NETWORK

The Ubicomp group and the 496 project team worked with UBC IT services to find a possible solution for this wireless connectivity problem. UBC IT staff provided some valuable information regarding UBC's wireless network.

4.1.1. UBC'S TWO WIRELESS NETWORKS

The UBC campus is covered by two wireless networks. One on the north side of the campus, and one of the south side of the campus. The border between the networks is around the Astronomy Building, just to the south of University Boulevard. The connectivity at this border is very poor. Whenever wireless users hop from the southern network to the northern network, they would be disconnected, and the user's wireless network card would need to drop the old network connection, detect the new network, and then establish a connection to the new network. Typically, this takes a significant amount of time and translates to long periods of non connectivity. The explained why during testing, very poor connectivity was experienced in areas near University Boulevard.

To prevent roaming on two bordering networks and experiencing delays in connectivity, it was determined that The Fugitive should only be played on one of the networks. This elimination of the troubled border area would help to reduce the amount of time that the game is disconnected from UBC wireless.

With two networks, the better wireless network should be chosen as the playing field for the game. UBC IT staff again helped with characterizing the Wifi connectivity of each network. They described the network on the north side of the campus as more established. This network has more access points, and future access points will continue to be installed. The network on the south side of the campus, on the other hand, is relatively new. A number of buildings in the region still do not have Wifi access points installed. Figure 2 on the next page shows the connectivity of UBC buildings in various parts of the UBC campus. Notice that in \ the left lower side of the map (northern campus between East Mall and West Mall), there are a significant number of wireless buildings that are positioned very densely together. This represents many access points and hence greater connectivity. On the other hand, the right side of the map (southern campus) is not as populated by live Wifi buildings. The distance between Wifi buildings are much farther apart. Therefore, to achieve better connectivity, the northern side of campus is chosen to be the new playing area.

4.1.2. PLAYABLE AREA WITH CONNECTIVITY INFORMATION

Following the decision to use the northern side of the UBC campus as The Fugitive's new playing area, a new map is to be drawn for the game. One issue that was identified during testing was that the limited connectivity awareness from the user's perspective. The user would not be able to know which areas guarantee the best wirelessly connection. See original map used by the game on page 12; notice that the map does not inform the user about connectivity data. If additional connectivity status could be provided by the map, the user can become more aware, and hence expect, accept, and work around encountered connectivity issues.

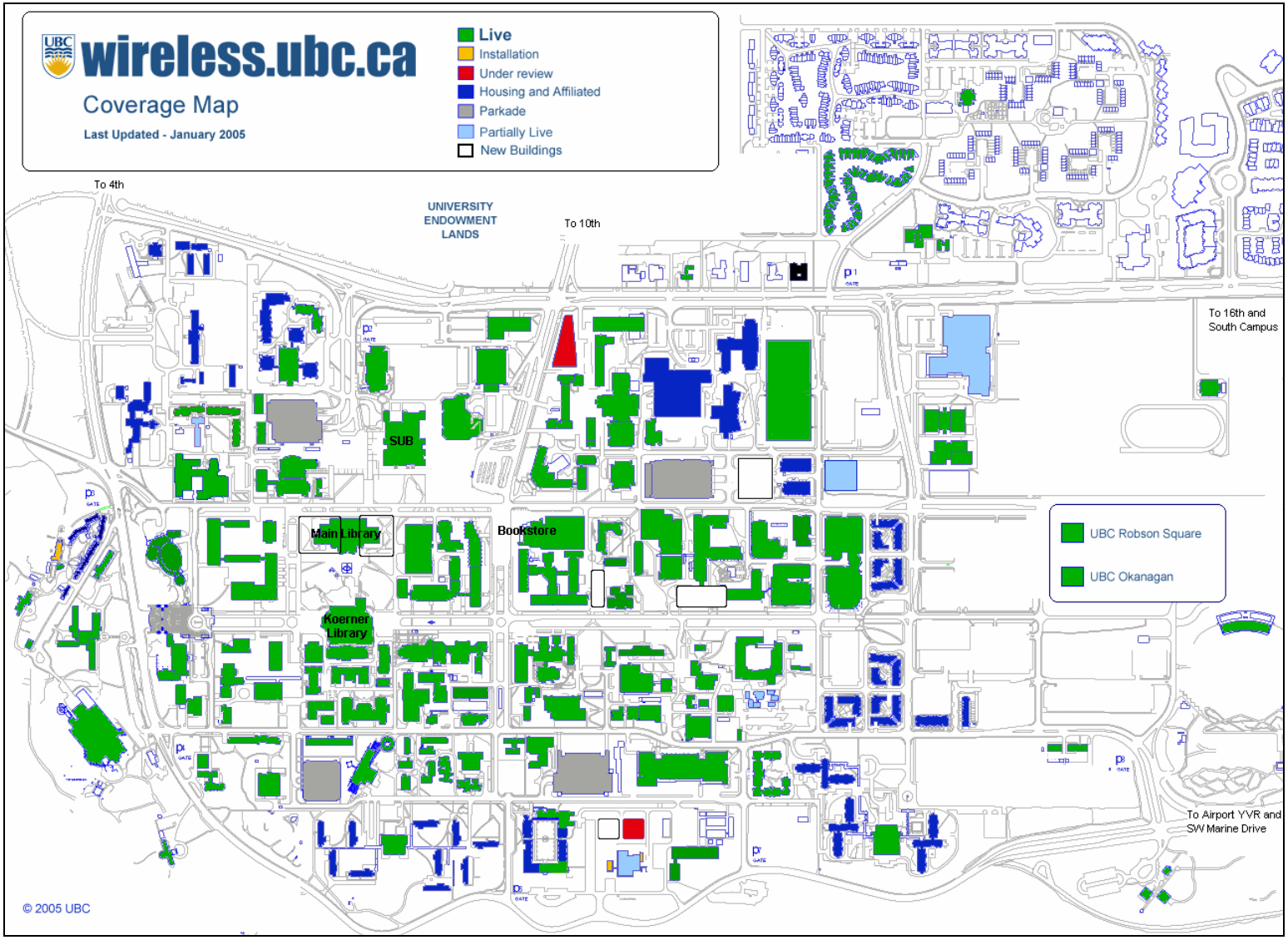


Figure 2. UBC Wireless Coverage Map [2]

Figure 3. The Fugitive - Original Playable Area

With this approach, further war-walking was done (with new positioning accuracy, and collecting location information specific to the new map). Areas with low connectivity were determined, and they are marked red on the game map to be indicated as problem areas. Also, a smaller playing field was used to allow the game to be played from start to end in the allotted 30-minute time. Any non-playable area on the map is marked with dark blue. The new map, as seen in figure 4, is on the following page. Notice that the size of the new playing area is less than a quarter of the original playing area. The new playable area was tested by the Ubicomp team as well as the 496 team, and it was determined to be an acceptable size to play the game in 30 minutes.

By changing the playing area to the north side of the campus and displaying connectivity information in the map, the game is now easier to play than before.

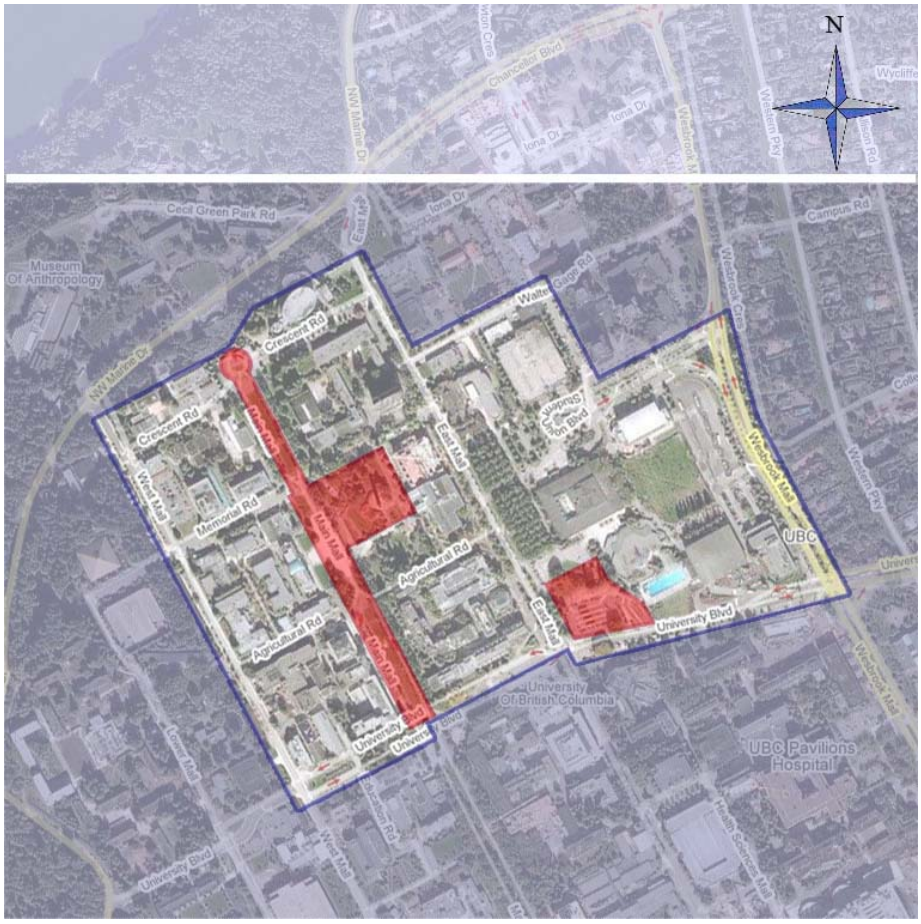


Figure 4. The Fugitive - Current Playable Area

4.2. UBC AUTOMATIC LOGIN

To alleviate the pain of manual login experienced by users every time the game is disconnected from the UBC wireless network, the implementation of an automatic login UBC is investigated to provide a viable solution.

4.2.1. ISSUES WITH AVAILABLE WIRELESS SOLUTIONS

A number of existing wireless solutions were initially analyzed and tested for the game. They included VPN and UBC Secure login. However, these attempts to provide automatic wireless login for the game were unsuccessful.

When connecting to the UBC wireless network using VPN, the VPN would maintain certain connection information, allowing the user to connect back to network if the period of disconnection is brief. However, if the time disconnected from the wireless network exceeds two to three minutes, the VPN tool is automatically shut down, and any maintained connection data is lost. Thereby, the user would still be required to manually login to the UBC wireless network.

Another solution was to log into the UBC Secure wireless network. UBC maintains two networks, one is a WEP network known as "UBC", and the other is a WPA network known as "UBC Secure". To connect to the WPA network, the user would only need to set up connection settings once. Following this setup, any connection to the UBC Secure network would be automatically done as long as there is wireless WPA network presence. However, this approach was also unsuccessful. The Place Lab library, used by the game to determine player location, is only compatible with a limited number of wireless cards. The compatible Avaya network cards used for testing the game were only capable of WEP connections. UBC Wireless IT also lent the project a number of Cisco WPA compatible wireless cards. However, some of these cards were not compatible with Place Lab, and of the ones that were compatible, none could connect to the UBC WPA network as they did not use the PEAP protocol required by UBC WPA connections.

4.2.2. AUTOMATIC LOGIN JAVA COMPONENT

Since available solutions to resolve the issue of UBC network login couldn't work, it made sense to develop a specialized one for the game. Therefore, an automatic login component was developed in Java and added to the game. When the game is being played, this component acts as an agent that automatically logs into the UBC wireless network using the player's CWL username and password whenever network connectivity is lost. If the component cannot log in, it continues its attempts at a period of every 10 seconds until a successful login is completed.

4.2.2.1. INITIAL IMPLEMENTATION

The automatic component to login to the wireless network was initially implemented using the following logic. The user would first provide his/her CWL username and password in a specified properties file for the game to use. When the game is in progress and the server

cannot be contacted, it can be safely assumed that the game is no longer connected to the wireless network. The program starts the automatic login thread and attempts to login to the UBC wireless network by issuing a formatted HTTP post request to http://login.wireless.ubc.ca:8090 with the CWL username and password of the user, as found in the specified properties file. If the post request is completed without any exceptions, then the game has successfully logged back into the wireless network. However, if any exceptions occurred, then it represents that the HTTP server cannot be contacted, and that the game still cannot detect the presence of any wireless network. The login thread will then sleep and try to login again after 10 seconds. The following logic diagram represents the implementation logic of the UBC automatic login.

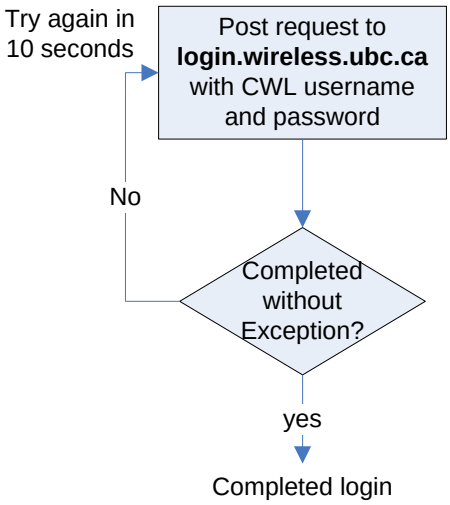


Figure 5. Logic Diagram for Initial Implementation of UBC Login

With a number of small tests over the course of two weeks, this simple component appeared to be functional and improved game playability significantly. Whenever the game was disconnected from the network, the user can simply walk around without modifying anything, and the game would soon be automatically connected again without the hassle of manual login.

4.2.2.2. CHALLENGES

When a complete game test was done during peak UBC Wifi access time (12 – 2PM)[3], using the newly implemented automatic login module, another connectivity issue was encountered. Around the Buchanan buildings on the north side of the UBC campus, the game was displaying non connectivity from the network for an extended period of time and could not synchronize with the server. Upon investigation, it was found that the game was falsely detecting the wireless network as not being connected when in fact it was. It appeared that every time that an HTTP post request was issued to the UBC login page http://login.wireless.ubc.ca, an exception was encountered. The automatic login component interpreted this as an unsuccessful login, and hence continued to try to log into the network every 10 seconds.

When this problem was further analyzed, it was found that the Buchanan area was producing a different Wifi login screen than other parts of the campus. At times of high wireless network traffic, any additional login requests from a user already logged onto the wireless network would be interpreted as a false request to maliciously flood the network traffic. Hence, the associated firewall at that login server would throw a "NoRoutetoHost" exception, indicating that access to the host server could not be granted. This exception was falsely interpreted as an error that translated to not being able to connect to the UBC wireless network, and the false interpretation hindered proper functionality of the game.

4.2.2.3. SOLUTIONS

To solve this exception problem, a new login implementation was designed to bypass the misinterpretation of this newly encountered error and any other similar future errors.

In the new implementation, the login component first sends an HTTP request to a random website to test whether the game is already connected to the internet. The random website to connect is chosen to be www.google.ca, as it seems to promise reasonable longevity for the near future. If the initial HTTP request is sent without any exceptions, then this means that the game can detect the presence of a wireless network and can sent internet requests.

However, at this point, it is still unknown whether the game is logged into the UBC wireless network. The login component then analyzes the response code for the issued HTTP request. If the response code is 302, meaning that the request was redirected to another HTTP page, then it means that the game is not yet logged into the UBC network. This is because when computers not logged into the UBC wireless network try to issue an HTTP request to any site, they are returned with a 302 redirect to go to the UBC wireless login site. Using this logic, a response code of 302 leads the component to proceed with the login, while any other response code at this stage represents that internet connectivity through UBC wireless has already been established, and the component login thread ends. Please refer to the following logic diagram.

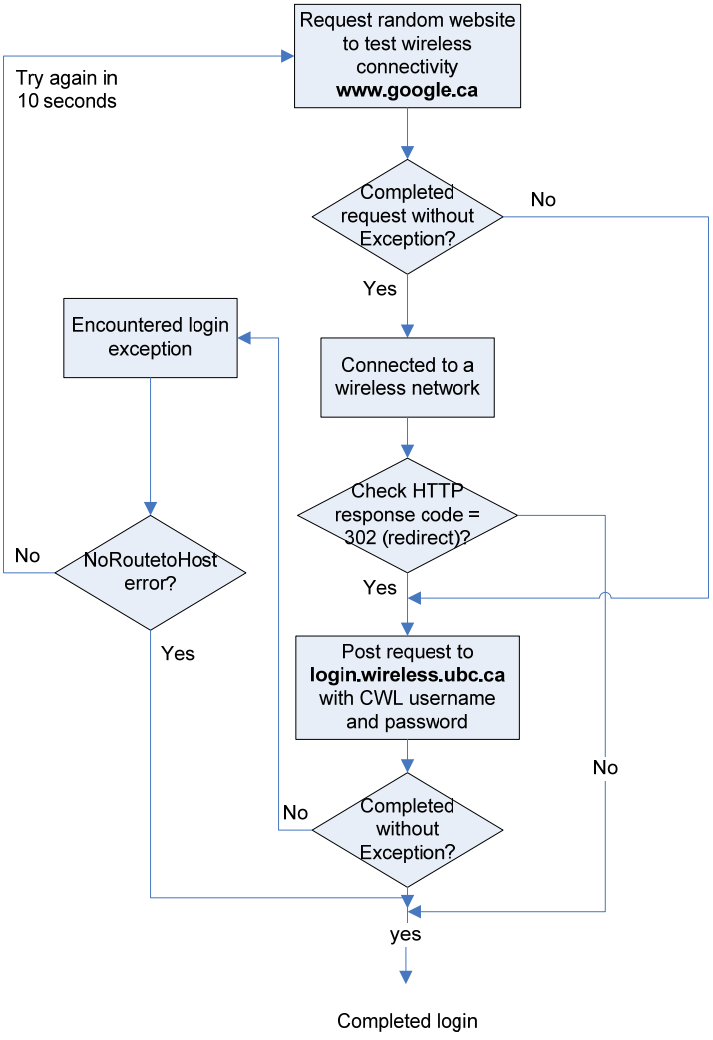


Figure 6. Logic Diagram for Improved Implementation of UBC Login

The rest of the login procedure is very similar to the initial implementation, where a HTTP post request is sent to the UBC wireless login page. The only addition to this portion of the program is to specifically address the error experienced at Buchanan. If a "NoRoutetoHost" exception is encountered, then the component program captures this exception and interprets the program as being already connected to the UBC wireless network. The component has been tested several times, and has been functioning very well in all tests.

4.3. RESULTING CONNECTIVITY IMPROVEMENTS

With a new playable area and the addition of the UBC login component, connectivity has been improved drastically. Previously, users suffered from long periods of connectivity failure and the pain of manually logging into the UBC wireless network. Now, these effects have been eliminated, if not minimized. Latest field tests have shown during a normal game of 30 minutes, the game was only disconnected for very little time. Frequently, almost immediately after the game was disconnected, it was connected again. Connectivity downtime was minimal and was not very noticeable. This helps to achieve the project's object of making the game functional and usable.

5. REPLAY TOOL

The replay tool was primarily modified by William Tsui, the 496 team partner for this project, and will be only mentioned briefly in this section to provide some general context information. The newly updated replay tool has many necessary features that the original tool lacked. Please refer to figure 7 below for a screenshot of the original replay tool. Compare this to figure 8, the screenshot of the updated replay tool.

The new features in the tool include ink stroke display, messages display via ink widget, the visibility of Bob throughout the game, the addition of colored ink ticks in the timer scroll bar to indicate ink message activity, and a new drop down menu of usability features such as rewind, fast forward, and play.

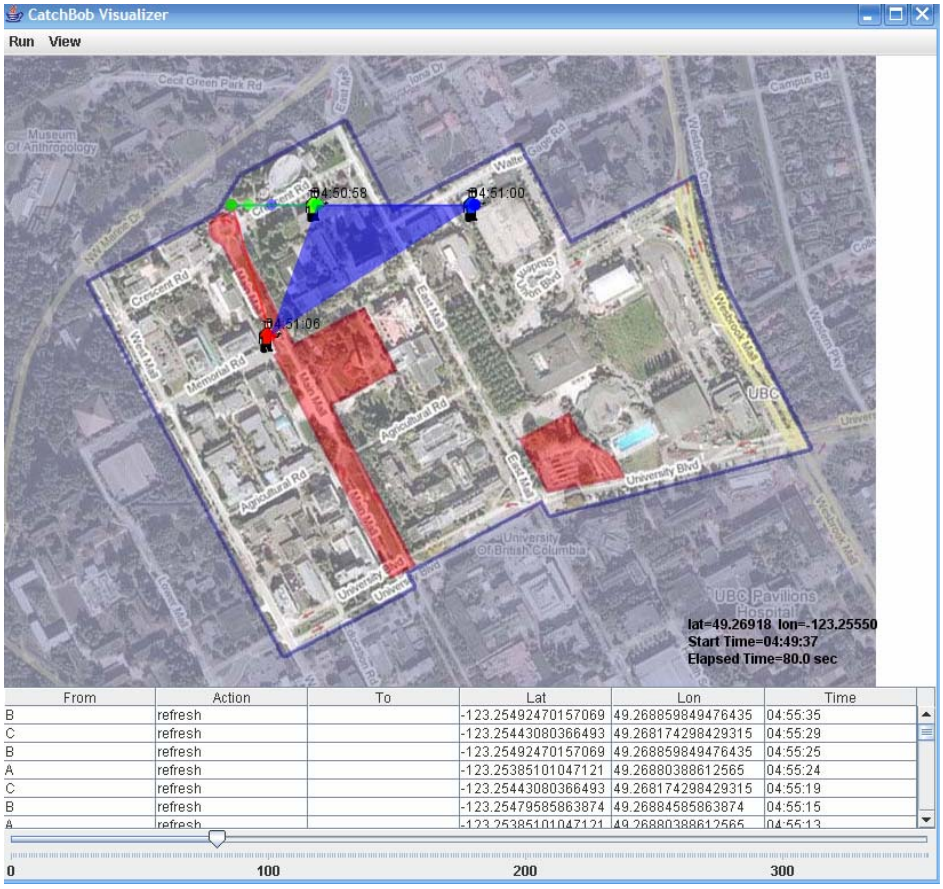


Figure 7. Original Replay Tool

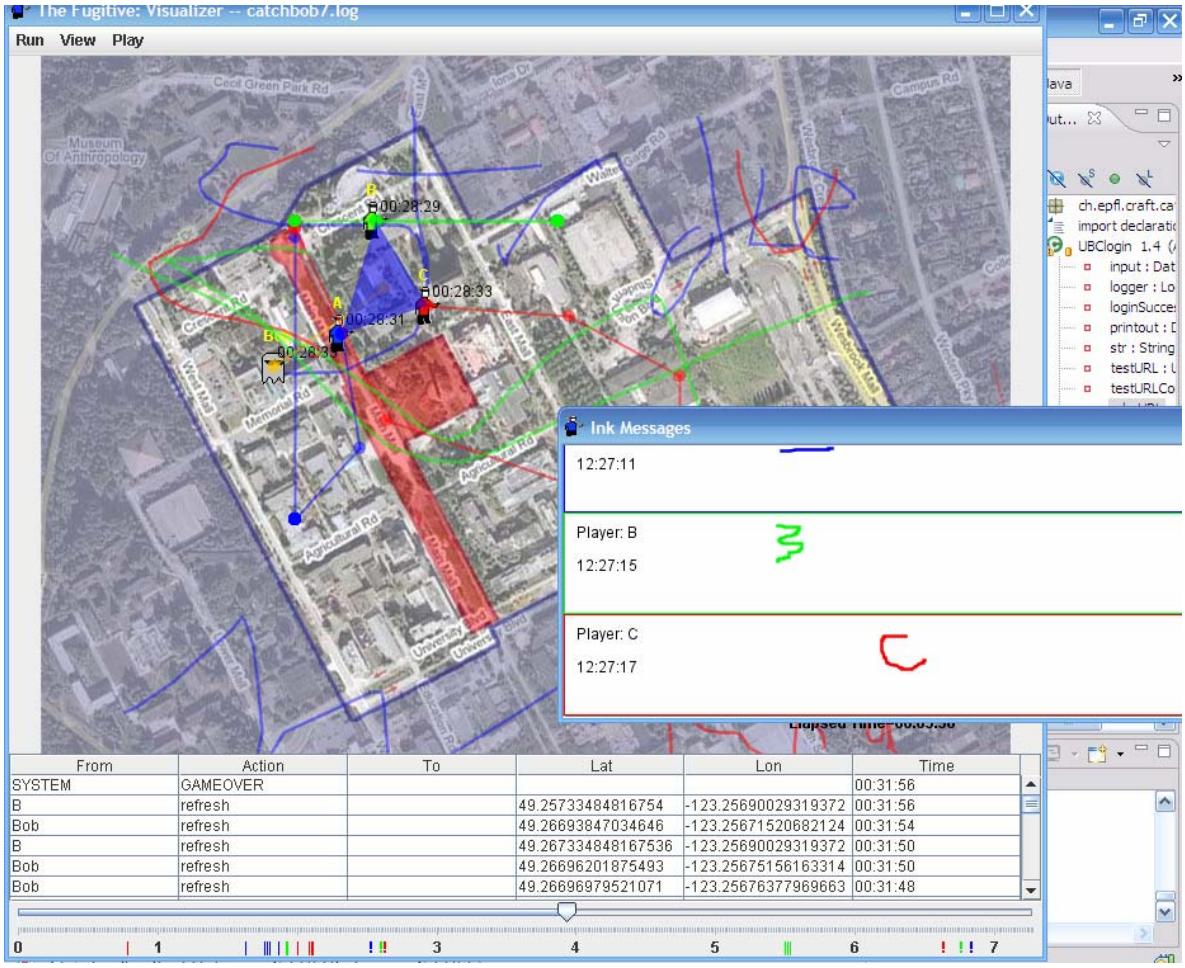


Figure 8. New Replay Tool

The addition of these features creates significant value for the Ubicomp group, as they can now use the tool to analyze game data for further understanding of strategy, collaboration, located-awareness research.

6. CLIENT USABILITY

To address the usability issues found during initial field testing, a number of tasks were performed to enhance the end user's game-play experience.

6.1. USER MANUAL

Since the Ubicomp plans to conduct a number of user tests once software development of The Fugitive is complete, a set of quick and simple instructions for new players to understand how to play the game is needed. A user manual was therefore developed to provide a brief 7-step guide to cover the objectives, features, and functionality of the game. Please see Appendix A for a copy of the user manual.

6.2. UI MODIFICATIONS

To improve the ease of use of the game, a small number of minor ser interface modifications were made.

6.2.1. SCROLL BAR

The testing and functional challenges experienced while playing game without a scroll bar were identified section 2 of the report. The scroll bar was a very simple and easy addition to the game, yet its value provided a lot of value during game-testing.

6.2.2. START OF GAME NOTIFICATION

The lack of a game start notification was identified by Phillip, a member of the UBIcomp group. Without the notification, new players would not intuitively know to synchronize their client game components with the server by clicking the server synchronization button at the start of the game. This is because there is nothing obvious that indicates the player to do so. Therefore, it is quite possible for new players to experience frustration and confusion, because they are not able to communicate to the rest of the team. UI changes were made to the map such that the initial

displayed explicitly states for players to synchronize with the server. Once this synchronization button is pressed, another bold message notifies the user that the game has started, and then the playable map used in the map is displayed.

6.3. GAME PARAMETERS

The game parameters of The Fugitive determine the speed and movement of Bob. However, as identified in section 2 of this report, the original game parameters made the game too difficult to play. These parameters allowed Bob to run very fast away from the players, and enabled Bob to detect players from far way. These parameters were modified to more reasonable values as determined through testing.

Bob's speed was lowered from 2.0 meters per second to 0.75 meters per second. Although the speed seems low at initial glance, it is actually very appropriate because the moving speed of players with tablet PCs is slower than normal walking speed. Also, Bob's safety distance was decreased from 150 meters to 100 meters. The catch distance and maximum distance between players for capturing Bob were both lowered to make the game easier and more enjoyable to play.

7. TESTING AND USABILITY RESULTS DISCUSSION

From continuous rounds of field tests and incremental iterative development of game features, playability of The Fugitive was improved. The primary objectives of improving positioning accuracy, obtaining reasonable continuous wireless connectivity, and implementing additional replay tool features were all met by the end of the project term. A number of small usability improvements were also made for the game.

The main challenge experienced in the project was to find an appropriate method of maintaining connectivity to the UBC wireless network. This took a number of different approaches, help from many parties, and a significant amount of testing to be successful. However, with this component now complete, the game is now easier to play and more functional than before.

Following this 496 project, it is expected that usability tests will start, and more project teams will be enlisted to assist with the iterative development method. Although this 496 project did not conduct in-depth usability testing, it would be interesting to see the results. A key suggestion for future teams working on The Fugitive is to start testing early and identify main challenges as soon as possible. This way, many different solutions can be implemented and tested.

8. CONCLUSION

This project focused on the testing, analysis, and modifications of The Fugitive, a location-aware game based on the UBC wireless network. Through much field testing, development changes, and component additions, the game now has better functionality and playability. The improvements come from increasing location accuracy, choosing a more playing area, maintaining more continuous wireless connectivity, adding features to the replay tool that enable research analysis, and tweaking UI at various parts of the game. Future development for this game will involve extensive usability testing in the ubiquitous computing research area and software modifications to further support that research. As final note, an important lesson learned through this project is to test for functionality and field behavior frequently so that technical challenges can be identified early such that there's enough time to find an appropriate solution.

REFERENCES

[1] Place Lab. Intel Research. March 2006, http://www.placelab.org/.

[2] UBC Wireless. UBC. March 2006. http://www.wireless.ubc.ca/coveragemap.html.

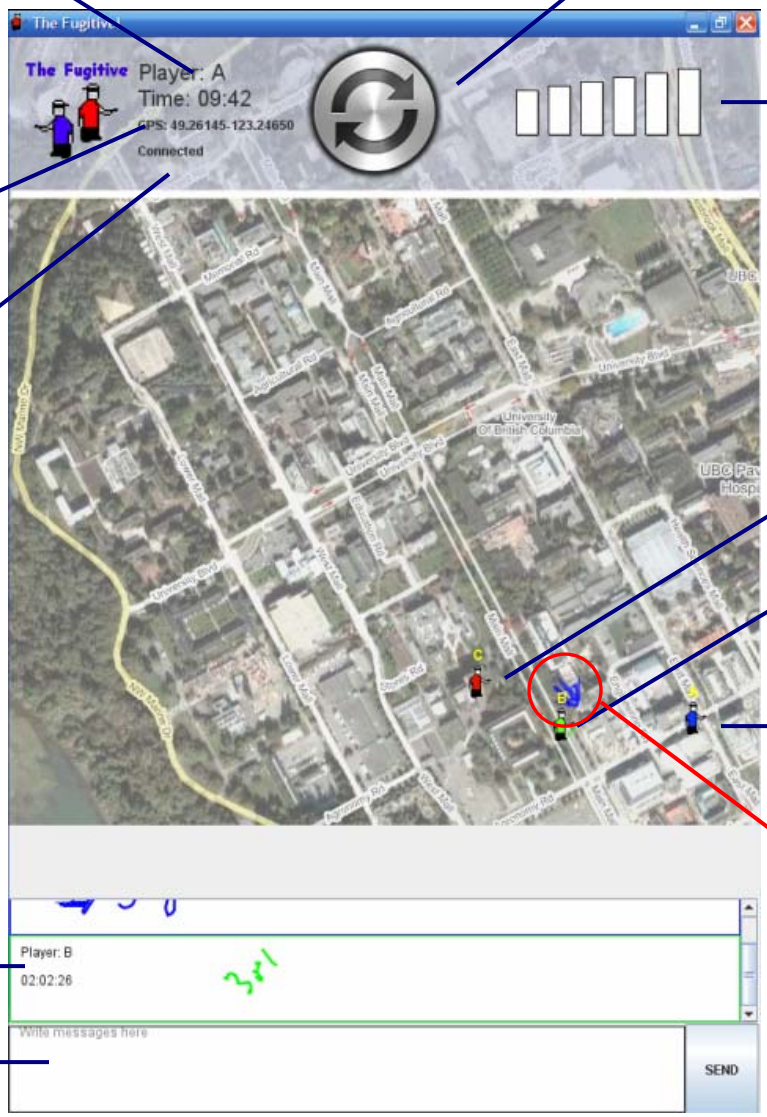
[3] Jonn Martell. IEEE. Deploying the world's largest campus IEEE 802.11b network. 2003.

APPENDIX A – USER MANUAL

The Fugitive – Location Based Wifi Game

User Manual

The Fugitive is a mobile location aware game that is based on the UBC Wifi network. The objective of this game is to work in a team of 3 players to capture a moving virtual person, Bob, by tracking and surrounding him.



Player Information
You are Player A, and the current game time is 9:42. You have 30 minutes to catch bob.

Synchronization button
Click to synchronize to server

Bob Proximity Bar
How close to Bob you are

GPS Location
GPS coordinates of your location based on the Wifi network around you.

Connectivity to server

UBC Campus Map

Can use ink to draw on map & communicate with other players.

Player C

Player B

Player position and ink messages are updated every 30 seconds

Player A
YOU ARE HERE

A drawn ink message

Public Scroll Area
History of all the ink messages that were written by the players, scroll up to see older messages

Ink Messaging Area
Ink area to write messages to other players: Player B and Player C

7 Steps to Play the Game

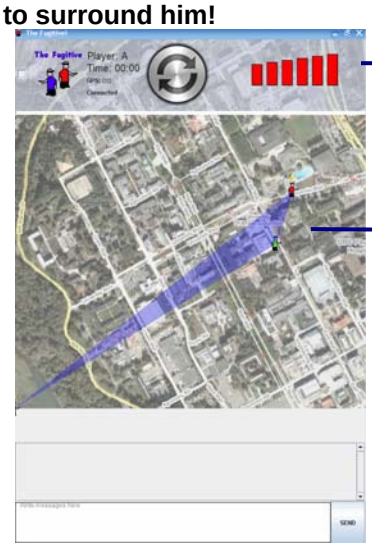
1. **Load the game**
   When the game is loaded, you should be able to see your own location at UBC in the map as shown above. The locations of the other players are in default positions.

2. **Press the synchronization button**
   To start playing the game and communicate with the other players (and to see the location of the other players if you're playing in the "view other players" mode), you must synchronize with the game server. The game is started when you connect by pressing the synchronization button.

3. **Start writing messages to the other players**
   You can now communicate to the other players by writing in the ink messaging area or drawing on the map. You can also read previous messages written in the public scroll area. When you write messages down, they appear in black ink. After they're transmitted to the other players, they change color depending on who wrote them (blue = player A, green = player B, red = player C). Messages drawn on the map will stay for around 5 minutes and then fade away.

4. **Move around to try to find Bob**
   In this first phase of the game, Bob is invisible and is hiding in a fixed location. So, you need to run around the UBC campus and keep on monitoring the Bob proximity bar. More bars will turn red as you get closer to Bob.

5. **Once you find Bob, get your team to surround him!**
   Bob can only be caught when your team surrounds him, or else he'll escape! Once you can get close enough to Bob, get your two other teammates to come close to you. If you all move close enough to Bob and he's somewhere between the three of you, a blue triangle will appear, indicating that you're on the right track. Bob is somewhere within that blue triangle.



6. **Once a triangle appears, indicating you have surrounded Bob, close in on Bob and trap him.**

Now that Bob is within the blue triangle, phase one of the game is complete. The game proceeds to phase two, where Bob is visible and can move. He will try to run away. You and your team must close in on him from all three points of triangle. If you can do it fast enough without Bob escaping, you'll trap him and win the game!

7. **You win! / You lose!**
   If you trap Bob within the allocated time frame, you'll win the game. However, if you can't trap him within this time, you'll lose. The default time to play the game is 30 minutes. Good luck!

APPENDIX B – UBC WIRELESS LOGIN COMPONENT

```
/**
 * Automatically logs into UBC Wireless with ubc CWL
 * login username and password from the .properties
 * file
 */
package ch.epfl.craft.catchbob.tablet;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.MalformedURLException;
import java.net.NoRouteToHostException;
import java.net.SocketException;
import java.net.SocketTimeoutException;
import java.net.URL;
import java.net.URLConnection;
import java.net.UnknownHostException;

import ch.epfl.craft.catchbob.CatchBobProperties;
import org.apache.log4j.Logger;
/**
 * @author Colleen Qin
 * Tests whether connected to internet, and if not, connect through UBC
wireless
 */
public class UBClogin {

  /**
   * @param args
   * @throws IOException
   */
  private URL url;
  private URL testURL;
  private URLConnection urlConn;
  private URLConnection testURLConn;
  private DataOutputStream printout;
  private DataInputStream input;
  private Logger logger;
  private String str;
  private boolean loginSuccess;

  /**
   * UBC Login constructor.  Creates the url needed for connection
   */
  UBClogin(){

    try {
      /* URL to connect log into UBC wireless */
      url = new
URL("https://login.wireless.ubc.ca:8090/goform/HtmlLoginRequest");

      /* Test URL to see whether the current login is working.
```

```
       * Default to use: google because it's unlikely it'll be out of
service anytime soon */
      testURL = new URL("http://www.google.ca:80/");
    } catch (MalformedURLException e) {
      e.printStackTrace();
    }

    logger = Logger.getLogger("ch.epfl.craft.catchbob.tablet");
    str = null;
  }

  /**
   * Try to connect to UBC wireless.
   * First test a url to see if connected to internet through UBC wireless.
   * It not, log in with UBC wireless username & password through a POST
request.
   */
  public void login() {
    loginSuccess = false; // Indicates whether login is successful

    while ( !loginSuccess )
    {
      /* Test a url connection to see whether connected to internet. */
      try {
        testURLConn = testURL.openConnection();  // Open test URL connection

        /* Set request properties */
        testURLConn.setDoInput(true);
        testURLConn.setDoOutput(true);
        testURLConn.setUseCaches(false);
        testURLConn.setRequestProperty("Accept", "*/*");
        testURLConn.setRequestProperty("Accept-Language", "en-us");
        testURLConn.setRequestProperty("Connection", "Keep-Alive");
        testURLConn.setRequestProperty("Host", "www.google.ca");

           /* List all the response headers from the server and check
            * if there is a redirect = 302.
            * If response code = 302 redirect, means that url was redirected
to login @
            * UBC wireless => therefore, not connected yet.
            * If response code was anything else (without exception), then
it indicates
            * that program is connected to the UBC wireless.
            * The first call to getHeaderFieldKey() will implicit send
            * the HTTP request to the server.*/
        boolean InternetOn = false;  // Variable to help track whethe there
is internet connection

        logger.info("Header Information: ");
        for (int i=0; ; i++)
        {
               String headerName = testURLConn.getHeaderFieldKey(i);
               String headerValue = testURLConn.getHeaderField(i);

               logger.info(headerName + "  " + headerValue);

               if (headerName == null && headerValue == null) // End of HTTP
response
               {
```

```
                  // No more headers
                  if( InternetOn ) // Had received other headers for HTTP
response
                  {
                     loginSuccess = true;
                     logger.info("Connected to internet");
                  }
                  break;
               }
               if (headerName == null) // First header of the HTTP response
               {
                  InternetOn = true; // Indicates that connected to the
internet
                              // but don't know whether connected through
UBC wireless yet

                  /* Check the HTTP response code to see whether it's 302
redirect */
                   String[] tmp = headerValue.split(" ");
               int responseCode = Integer.parseInt(tmp[1]);
               logger.info("HTTP response code = " + responseCode);
               if( responseCode == 302 )  // redirect
                break;
                }
            }

         /* If arrived here & loginSuccess = true, then wireless internet is
connected.
          * No need to do rest. */
         if(loginSuccess)
           break;

      } catch (IOException e2) {
         e2.printStackTrace();
      }

      /* Connect to UBC wireless by issuing a http POST request with CWL
       * log in username & password */
      try {
         logger.info("Attempting to log into UBC wireless");
         urlConn = url.openConnection();

         /* Set the content information to be sent through the HTTP request
*/
         urlConn.setDoInput(true);
         urlConn.setDoOutput(true);
         urlConn.setUseCaches(false); // No caching
         // Specify the content type.
         urlConn.setRequestProperty("Content-Type", "application/x-www-form-
urlencoded");
         urlConn.setRequestProperty(
                  "Accept",
                  "mage/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-
powerpoint, application/msword, */*");
         urlConn.setRequestProperty("Referer",
"https://login.wireless.ubc.ca:8090/index.asp");
         urlConn.setRequestProperty("Accept-Language", "en-us");
         urlConn.setRequestProperty("Connection", "Keep-Alive");
```

```
        urlConn.setRequestProperty("Host", "login.wireless.ubc.ca:8090");

        Integer loginLength = new
Integer(CatchBobProperties.ubcUsername().length() +
CatchBobProperties.ubcPassword().length());
        loginLength += 31; // Add the length of the UBC login default text
        urlConn.setRequestProperty("Content-length",
loginLength.toString());
        urlConn.setRequestProperty("Cache-Control", "no-cache");

        // Send POST output.
        printout = new DataOutputStream(urlConn.getOutputStream());

        /* Get username and password from CatchBobProperties */
        String content =   "username=" + CatchBobProperties.ubcUsername() +
                    "&password=" + CatchBobProperties.ubcPassword() +
                    "&login=Login";
        printout.writeBytes(content);
        printout.flush();
        printout.close();

        // Get response data.
        input = new DataInputStream(urlConn.getInputStream());

        while (null != ((str = input.readLine())))
        {
          logger.info(str);  // Output the login information to indicate
success
        }
        input.close();
        loginSuccess = true;
        logger.info("UBC wireless connected");

      }
      /* Catch exception being thrown at login.ubc.wireless.ca for pinging
too much */
      catch (NoRouteToHostException e1) {
        // Means already connected to UBC wireless
        loginSuccess = true;
        logger.info("Reached UBC login page, but NoRouteToHostException.");
        logger.info("UBC wireless already connected");

        /* Print out IP address at this time - for debugging reasons*/
        try {
          InetAddress addr = InetAddress.getLocalHost();
          logger.info("Current IP address = " + addr.toString());
        } catch (UnknownHostException e) {
          e.printStackTrace();
        }

        break;
      }
      /* Catch socket timeout exception encountered during testing */
      catch (SocketTimeoutException e2) {
        logger.info("Socket time out exception");
        logger.info(e2);
        try {
          Thread.sleep(10000);  // Sleep for 10 seconds
        } catch (InterruptedException e1) {
```

```
            e1.printStackTrace();
          }
        }
        catch (SocketException e3) {
          logger.info("Socket exception from program.");
          logger.info(e3);
          try {
            Thread.sleep(10000);  // Sleep for 10 seconds
          } catch (InterruptedException e1) {
            e1.printStackTrace();
          }
        }
        catch (IOException e) {
          /* Cannot log into the UBC wireless netwok. Hence, let thread sleep
           * for 10 seconds and then try to log in again. */
          logger.info("IO exception from program.");
          logger.info(e);
          try {
            Thread.sleep(10000);  // Sleep for 10 seconds
          } catch (InterruptedException e1) {
            e1.printStackTrace();
          }
        }
      }
    }

}
```