Design and Implementation of Walking Incentive Game Base on Cloud Service

Liwei Fu¹, Hirohide Haga^{2,*}

¹Origami Accounting, 4500 10230, Jasper Avenue, Edmonton, Alberta, T5J 4P6, Canada

²Graduate School of Science and Engineering, Doshisha University, 1-3, Miyakotani, Tatara, Kyotanabe, 610-0321, Japan

Copyright ©2015 by authors, all rights reserved. Authors agree that this article remains permanently open access under the terms of the Creative Commons Attribution License 4.0 International License

Abstract In this article, we focus on the proposal, design, and implementation of a mobile game for walking. Walking is a simple, unobtrusive, and effective sport that can help people stay healthy. Researchers have proposed that games, rather than being simply entertaining, can provide incentive to accomplish work that people are less willing to do. Such a game is called a “serious game.” In our proposed game, the player’s walking steps in the real world are counted as the unique resource and are transformed into their game character’s achievement, called the territory. From an implementation point of view, we chose the iOS platform and Windows Azure cloud service as the front-end and back-end, respectively. On the iOS platform, a pedometer based on the iPhone’s three-dimensional accelerator was developed to count the players’ steps. In addition, Google Maps was used to represent the game situation. Azure was applied as a server to enable multi-user and interaction features. The cloud server maintains low development cost, as well as efficient usability, stability, and scalability.

Keywords Serious Game, Healthcare, Walking, Mobile Device, Cloud Service, iOS, Windows Azure

Walking is uniquely beneficial in that people can walk while relaxing, chatting, or thinking, yet still obtain the benefits of healthy exercise. Walking a certain amount of steps at a specific speed that is a little faster than normal also boosts immunity. However, because of the conveniences of technology and engineering, we seldom walk if we do not want to. Automobiles are the more preferable choice even when destinations are close enough to be reached on foot.

Gaming has a long history as human entertainment. Computer and electronic technology provide a new gaming stage to enjoy and to interact with other players. The concept of using games for purposes other than entertainment, so-called “serious games[1],” has been suggested. Games can be used as a tool to incentivize work that people are less willing to do or that computers have little ability to do.

The effectiveness and simplicity of walking as a sport with the additional incentive effect of electronic gaming makes a healthcare game that encourages people to walk more – and thus keep healthy – very attractive. We therefore propose a game designed to increase people’s willingness to walk. This walking incentive game runs on an iOS and Windows Azure platform. The design and implementation of the game and some corresponding performance and function test results are presented in the following sections.

1 Introduction

We propose, design, and implement a walking incentive mobile device game to promote human health. This walking incentive game counts a player’s walking steps and uses the number of steps as the sole resource. Performance is determined by the amount of walking a player does. With some multi-user-featured game rules, social elements are introduced that enable players to interact and compete. The ultimate goal is that players are encouraged to walk, resulting in better health.

Walking is a very common human activity that is simple, unobtrusive, and effective physical exercise. Unlike other sports, such as badminton or baseball, walking requires no equipment, arena, or skill and is time effi-

2 System Design

The design of our walking incentive game is inspired by “capture the flag.”[2] “Capture the flag” is a traditional two-group game in which the aim is to steal the opposing team’s flag. The first group to capture the other’s flag, while preventing their own flag from being captured, wins. Various barriers and member rules, such as capturing opposing team members, can be applied to make the game more interesting. A mixed-reality version application of capture the flag is available for smartphones. With smartphones, the players physically role-play virtual characters who try to capture enemy flags by traversing different landscapes, which establishes a

direct and real-time link between the real and virtual worlds.

2.1 Components

The components in our proposed game are “Knights,” “Mercenaries,” “Territories,” and “Gold.”

- **Knights:** The knight is the main character. The knight’s task is to recruit mercenaries and acquire territories. In order to defend territories, the knight must lead his mercenary army to build castles every time a territory is occupied. In some cases, the knight and his army will invade another player’s territory when two players collide over a territory.
- **Mercenaries:** The mercenary is the only resource in the game. Mercenaries are recruited by the knight and are able to build castles and fight. In our walking incentive game, the number of mercenaries depends on the number of steps a player takes. The more steps the player walks in the real world, the more mercenaries his knight can recruit in the game.
- **Territories:** The game’s territory is a circular ground area in the real world. The center position and radius of the territory are determined, respectively, by the location where the player finishes his/her thousandth step and the corresponding actual steps taken at that time. For example, if a player completes his 1,000th step in front of a bus stop, his territory would be the 10-meter radius round area surrounding that bus stop. The player then continues walking and achieves his/her second territory after 2,000 steps. The range of this second territory would be a 20-meter radius with a center at the 2,000th step’s geographical location.
- **Castles:** Castles are built at the center of territories for defense purposes. After building a castle, the knight and his army will leave because of the continued walking of the player in the real world. Once another player’s knight and army invade, the castle will role-play as the defender to fight against the enemy knight.
- **Gold:** Gold is awarded to the battle winner after a battle. Currently, the only usage for this gold is to promote the knight’s level. Other functions, such as purchasing game tools or territories, have not yet been explored and will be considered in future work.

The basic time unit of this game is daytime. In order to encourage players to walk every day, rather than just for a few days, we introduced a character level mechanism. Knights must gain more experience to promote their levels, which allows them to perform better in castle building and battle, thus making them more likely to win controversial territories. The only way to gain experience is to walk more and play the game daily. At the end of a time unit, all the game data, except for the knight’s level, is truncated. In other words, at the

end of each day, armies, territories, castles, and gold are converted to experience. Every day when the game begins, all players are at the same level, with zero armies, territories, castles, and gold. The only differences that remain are their knight’s level and corresponding experience.

2.2 Collision Detection and Processing

Since the game is designed for multiple users, it is highly possible that collisions over territories will occur between players. This is actually a key feature of the game, enabling more interaction and creating more excitement. Collision detection and processing is an important issue in the game’s design and implementation. The territory model used in this game is round, which is simplest.

As previously described, a player obtains a territory at each one thousand steps. This territory is temporary and needs to be checked for potential overlap with other territories. The game system searches for nearby territories that belong to other players. If the new territory is included in any other player’s territory (whether as a part or a whole), the two players (player **A** and player **B**) must battle. Player **A** is the new, temporary territory’s owner, while player **B** is the owner of the pre-existing territory. The battle participants are player **A**’s knight and army as the offender and player **B**’s castle as the defender

Figure 1. Territory collision between player **A** and player **B**

The battle system determines the result of a battle based on the offender’s knight’s level and number of mercenaries, and the defender’s defense, which is determined by the castle builder’s knight’s level and mercenary number. The loser of the battle gives up the controversial territory by reducing his/her territory radius, as shown in Figure 2 (assuming loss by player **A**). The winner either gains the new territory or retains the pre-owned territory and some receiving a bonus of gold.

2.3 Determining Game Parameters

There are several important parameters in this game, including the number of mercenaries, knight level, experience, castle defense, and gold. All of these parameters are directly or indirectly related to the number of steps walked by the player. To determine the availability of

Figure 2. Result of battle between player **A** and player **B**

software implementation and post-implementation performance tests, as well as to provide a deep comprehension of the game rules, the numerical definitions of these vital parameters are provided below.

2.3.1 Number of Mercenaries and Territory Radius

The number of mercenaries and territory radius are directly related to the number of steps. Each time a player completes 10 steps, his/her knight character gains one mercenary. There is no limitation to the number of mercenaries that a knight can recruit in a single day. The territory's radius is one hundredth of the step number with a metric length unit. Since a round 200-meter radius area is, in reality, rather large, we set this as the ceiling for territory radius. These two relations are expressed by the following equations, where army represents the number of mercenaries:

$$army = \frac{step}{10} \quad (1)$$

$$radius = \begin{cases} \frac{step}{100} & step \leq 20,000 \\ 200 & step > 20,000 \end{cases} \quad (2)$$

2.3.2 Attack and Defense

The defense of a castle can highly impact the success of a battle. The defense depends on the size of the army and knight level when the castle is being built. In order to reasonably express the influence of knight level, it is amplified 50 times:

$$defense = army + 50 \times level \quad (3)$$

The attack of a knight-led army is defined similarly:

$$attack = army + 50 \times level \quad (4)$$

Once the offender's attack and defender's defense are established, the outcome of the battle can, for the most part, be predicted. However, a battle participant with a higher attack or defense is not always the winner. The result is probability driven; therefore, the participant with the higher attack or defense numerical value has a higher possibility of winning the battle, but could still lose.

$$P_{attack} = \frac{attack}{attack + defense} \quad (5)$$

$$P_{defense} = \frac{defense}{attack + defense} \quad (6)$$

This approach is intended to encourage low-level players because they will always have the possibility of winning even if their enemy is a high-level knight or a castle built by a high-level knight. Moreover, as long as they walk enough steps, their knights can recruit a large number of mercenaries, which means they have a greater advantage when up against a high-level knight with a small number of mercenaries.

2.3.3 Gold Bonus

It is a good approach to balance the game system in terms of knight experience and knight level. The principles of providing a gold bonus include: award more gold to the winner when the loser has a higher knight level, which promotes the character of high-level knights; award more gold to the winner when there is a wider difference between the probabilities for the winner and loser, which encourages players to walk more to increase their probabilities; halve the gold award if the winner has a lower result probability to reduce the influence of luck. Based on these principles, the gold bonus equation is as follows, where all the probabilities are prior probabilities calculated by the above equations. For example, P_w and P_l represent the final winner's and loser's result probability based on equation (5) or equation (6), respectively, and l_l indicates the "loser's level."

$$G = \begin{cases} \frac{1}{2} \times k \times l_l \times (P_l - P_w), & P_w < P_l, \\ k \times l_l \times (P_l - P_w), & P_w \geq P_l \end{cases} \quad (7)$$

Parameter k is the amplification constant, which gives the gold an influence on knight experience that is equivalent to the other factors. Tentatively, k is set to 100.

2.3.4 Experience

The experience and knight level parameters influence the game's balance directly. At the end of every day, the data for the three parameters of gold, territory, and army size are transformed into experience. Both army and territory size are reflections of the player's steps, but we only take territory size into consideration for experience. Assuming that G_m represents the gold bonus for m battle and S_n is the area of n territory, the daily experience a knight can obtain is:

$$experience = \sum G_m + \sum \frac{S_n}{c} \quad (8)$$

Parameter c is a contraction constant that reduces the territory area. Tentatively, c is set to 10,000.

Both k and c are parameters that make the experience numerically reasonable. Based on equation (8), despite the gold income and territory loss as a result of battles, a player's daily experience income can be listed by

his/her number of steps, as shown in Table 1. For example, a 5,000-steps player would have five territories with radiuses of 10 to 50 meters. The sum area of the five territories divided by c is the player's experience, which is around 1.73.

Table 1. Amount of walking and experience income

Step number	Experience
1,000	0.03
2,000	0.15
5,000	1.73
10,000	12
15,000	39
20,000	90

Actual experience is very limited if a player walks less than 10,000 steps a day, which supports the main purpose of this walking incentive game. A player's game performance is acceptable only if the player walks more than 10,000 steps each day.

2.3.5 Knight level

The last, but not least, parameter that needs to be determined is knight level. How much experience is necessary to promote a knight's level is an important issue. If it is too easy for a knight to achieve a high level, based on equations (3) and (4), army attacks and castle defenses would be dramatically increased, which can create an imbalance in the game. On the other hand, if it is too difficult to reach a higher level in the game, player activity will be significantly influenced. Remaining a low-level character despite long-term game play is not encouraging. Consider the following quartic polynomial (9). The corresponding relationship between experience and level is listed in Table 2. Note that parameter l in equations (9) and (10) means the "level."

$$experience = l^4 + 2 \times l^3 + 3 \times l^2 + 4 \times l + 5 \quad (9)$$

Table 2. Each level's experience requirement based on equation (9)

Level	Experience required
1	15
2	57
3	179
4	453
5	975
6	1865
7	3267
8	5349
9	8303
10	12345

Table 3. Each level's experience requirement based on equation (10)

Level	Experience required
1	10
2	26
3	58
4	112
5	194
6	310
7	466
8	668
9	922
10	1234

It is obvious that this level system is too difficult when considering the actual experience a player receives,

Figure 3. Typical game scenario

which is listed in Table 1. With a cubic polynomial, things are much better (see equation (10) and Table 3).

$$experience = l^3 + 2 \times l^2 + 3 \times l + 4 \quad (10)$$

This makes it easier for a player to reach higher levels. The difference can reach up to 10 times when a knight is about to reach level 10 compared to the former scheme. We therefore adopt this more reasonable level scheme.

2.4 Typical Game Scenario

In this section, we describe a typical game scenario in a game unit in order to illustrate the rules more clearly. Suppose that a player **A** with a level 2 knight enters the game at 8 a.m., the starting time of every game unit. Player **A** is a student who is about to travel to school on foot. He covered 6,000 steps and achieved six territories during his trip to school. The first six territories were achieved smoothly without any battles. When he reaches 7,000 steps, he will receive a territory with a 70-meter radius. However, as shown in Figure 3, a portion of his temporary territory is located within the 100-meter radius range of another player's territory—player **B**—who has a knight level of level 3. The detailed information for these two battle participants is shown in Table 4.

In Table 4, the size of the two players' armies are determined by their territory range when the battle takes place. Their attack and defense are calculated by equations (3) and (4). Player **A**'s chance of winning the battle is approximately 41%. However, after determining the game server, player **A** won the battle and received a gold bonus of 27. Player **B**'s territory radius was reduced to the point where there is no overlap with player **A**'s territory, as shown in Figure 4. Player **B** also lost some territory that did not overlap with player **A** as a compromise to implementation complexity, because reducing the radius is the easiest way to deal with player overlap.

Eight p.m. marks the end of the game for the day. Suppose that player **A** walked more than 14,000 steps in total and received 13 territories. He engaged in six battles, two of which he lost. As a result of one of these losses, he failed to achieve a territory because the center of his territory was within the winner's territory. The total area of his 13 territories is approximately 282,000 square meters. In addition, he received a bonus of 160

Table 4. Battle participants' detail information

	Player A	Player B
Knight level	2	3
Army scale (mercenaries)	700	1000
Battle unit	Knight and army	Castle built by the knight and army
Battle parameter	Attack=800	Defense = 1,500
Win probability	41%	59%

Figure 4. Player B lost with reducing his territory radius**Table 5.** Summary of daily game performance

Total game time	12 hours
Total walking steps	14259
Territory number	13
Total territory area	282,000
Battle won over total	4 over 6
Total gold achieved	110
Total experience transformed	188

gold units in the course of his four wins. Table 5 is a summary of his game performance.

Experience is not erased, even after it has been used to promote a player's knight's level. Assuming that player A has previously had 40 experiences, his current total experience would be 228, giving his knight a level of 5. At the beginning of the next day's game, he will have only a level 5 knight with 228 experience points.

3 Implementation

3.1 System architecture

Based on the game rule design described above, it becomes clear that our walking incentive game system should have two basic modules: one for local functions, such as the pedometer, location determination, and user interface, and the other as a game server with a multi-user feature. We chose the iOS[3] platform and Windows Azure[4] cloud service. iOS devices such as the iPhone have many fancy features that are very suitable to this game. Windows Azure is used to replace conventional servers due to its cloud features, so developers do not need to spend much time on server building. The iOS

application is responsible for counting steps, map view display, and user interface. The Windows Azure service provides collision detection, processing, and data storage.

3.2 Implementation of iOS application

In the design and implementation of an iOS application, developers usually follow the Model View Controller (MVC) design pattern. MVC is a high-level pattern that classifies objects according to the general roles they play in an application. Objects are classified into three types: model objects, view objects, and controller objects. A major concern in application design is choosing or creating objects that fall into one of these three categories. Each type is separated from the others by abstract boundaries and communicates with other object types across these boundaries. By adapting the MVC pattern, object-oriented programs can be benefited in several ways. Objects tend to be more reusable and their interfaces better defined. Programs overall are more adaptable and extensible[5]. We therefore followed the MVC design pattern for this game's iOS application design.

Figure 5 shows the structure of our iOS application. It contains four controller objects (shown as round rectangles), three view objects (shown as ovals), and two model objects (shown as HDs). Among the four controllers, the root is `UITabBarController` (left-most rectangle), which is a special type of controller in iOS that enables users to swap between several views. There is no actual view for `UITabBarController` (top center); instead, it maintains an array of view controllers that control views, which are the other three controller objects in this application. It also maintains a tab bar at the bottom of the screen with a tab for each view controller in the array.

The three views are responsible for displaying the game profile, pedometer information, and map view that contains the territory information. Thus, the view controllers should have the ability to access the game server model, the pedometer model, or both. The following sections in this chapter will focus on the implementation of the pedometer model and map view controller, as they are the most significant modules and require the most development.

3.2.1 Pedometer

The player's number of walking steps is the game's only resource. It affects a player's game performance significantly. Therefore, the pedometer algorithm de-

Figure 5. iOS Application Structure

Figure 6. Delegate relationship

sign and its implementation are fundamental. iOS SDK (Software Development Kit) packs iPhone's accelerometer feature in a class called `UIAccelerometer`. The pedometer model object can be registered as a delegate and receive acceleration related data from the on-board accelerometer. Once the device's acceleration changes, a method called `accelerometer:didAccelerate:` is invoked (shown in Figure 6).

The pedometer algorithm is based on an acceleration motion vector. When a human walks, body acceleration changes dramatically. Suppose a human has just finished taking a step with his left leg and is about to start another step with his right leg. Figure 7 details the four stages of this step. The acceleration vectors in Stage.1 and Stage.2 have upside components, while those in Stage.3 and Stage.4 have downside components. Therefore, in one step, there must be two moments where the angle between those two corresponding acceleration motion vectors is large enough to be detected and recognized. Thus, the solution is obvious. If we can sample the acceleration of the iPhone at a specific frequency, say $1/\Delta t$, the angle between two contiguous motion vectors, say θ , can be calculated. An angle θ greater than the threshold can be considered a step.

The acceleration vector obtained from the `accelerometer:didAccelerate:` method is three dimensional. Suppose two continuous vectors are:

$$\vec{x}_1 = (x_1, y_1, z_1) \quad (11)$$

$$\vec{x}_2 = (x_2, y_2, z_2) \quad (12)$$

The cosine of the angle θ between them is:

$$\cos \theta = \frac{\vec{x}_1 \bullet \vec{x}_2}{|\vec{x}_1| \cdot |\vec{x}_2|} \quad (13)$$

Figure 7. Accelerations in different stages of one step

Note that " $\vec{x} \bullet \vec{y}$ " is the inner product of two vectors $\vec{x}$ and $\vec{y}$, and $|\vec{x}|$ is the norm of vector $\vec{x}$.

The only problems are how to determine time interval Δt and the threshold of angle θ . Since there is not enough supporting research for reference, we can only estimate these by experience. If the estimation of Δt is too large to be a reasonable value, the user may have finished more than one step during that period. If it is too small, the angle between the two contiguous vectors may not be large enough. Considering that an adult's walking speed is usually less than 10 km/h and step length is around 75 cm, the time interval should be greater than 0.27 s. In addition, the walking speed should be fast enough to be beneficial. Finally, we set the time interval Δt at 0.3 s, which means that the detection of walking faster than 3.3 steps per second or slower than 1.6 steps per second is not guaranteed. For angle threshold θ , 35 degrees seems to be an acceptable value based on practical experiments.

3.2.2 Map View Controller

There are several technologies available on the iPhone to determine a user's geographical location. They include GPS (Global Positioning System), GNASS (Global Navigation Satellite System), Wi-Fi assisted location[6], and cellular assisted location. The first two are satellite based. The Wi-Fi assisted location is a cloud-computing technology. The device's location (acquired by other location technologies, such as GPS) and the Wi-Fi data, including SSID (Service Set Identifier) and router information (such as a MAC (Media Access Control) address), are uploaded and stored in the service provider's server.

Google released Google Maps SDK for iOS at the end of 2012[7], which makes it possible for iOS applications running on iOS to use Google Maps. We integrated Google Maps in the implementation of our walking incentive game to improve the map view experience. This makes programming and debugging more complex because the Apple Maps service is originally integrated while Google Maps is provided by external SDK. A comparison of Google Maps and Apple Maps is provided in Figure 8.

Figure 8. Comparison between Google Maps and Apple Maps

3.3 Implementation of Windows Azure Game Server

3.3.1 Overview

Our prototype system was developed with Windows Azure[4]. Windows Azure is Microsoft's cloud computing platform and infrastructure for building, deploying, and managing applications and services through a global network of datacenter managed by Microsoft. Azure Mobile Services provides backend features to applications running on mobile devices. Our walking incentive game requires data storage on a server, player authentication, and notification when achieving or losing territory, which can all be provided by Azure Mobile Services. CDN (Content Delivery Network) is a caching mechanism that accelerates access to the datacenter's stored content. It has dozens of sites around the world, each capable of storing copies of Windows Azure data. The first time a user accesses a particular data unit, the information it contains is copied from a Windows Azure datacenter into local CDN storage in that geographic area. After this initial access, accesses in that region will use this copy and get faster access.

In the proposed game's implementation, we use Azure Mobile Services as the cloud server. The game rule logic is performed in server-side codes. The user authentication feature is implemented by identity using Facebook. In addition, notifications about a player's game performance can be pushed to the device when necessary.

3.3.2 Primary Game Rule Implementation

In order to perform the game rules on the server side, we first have to define the format of the uploaded message. Territory location should be uploaded as latitude and longitude. The territory's radius should also be provided so that the server can detect whether there are any overlap collisions with other players. If the answer is positive, the server will deal with the battle, which will require the knight level and army size of the two participants. After determination of a territory, all players who were ever involved in related battles would be notified, which means that a player's name should also

be a parameter in the storage table. Moreover, according to Apple's push notification service, a device's token should be provided. Each data entity is processed and then stored in Windows Azure as a table row.

Every time a player completes a thousand steps and applies for a territory, his device sends a message in specific format to Windows Azure Mobile Services. After receiving a territory request, the server then checks whether the request overlaps existing territories. We simply read all the entities in the territory table and check to determine if any potential collision exists. Better methods to do this checking exist, such as dividing the game world into grids so that the server only has to check those territories located in the same grid as the request. As this game is just a prototype, the number of players should not be too large; therefore, the table traversal is acceptable and more practical. The problem, however, is how to we calculate the distance between two territories by their latitudes and longitudes. As the earth can be regarded as a sphere, it becomes a simple mathematic issue: the calculation of distance on a sphere's surface. In fact, the earth is actually an ellipsoid; however, we do not need too much accuracy for this calculation since the location accuracy is only on a meter level. The earth surface distance can be calculated as described below.

In iOS SDK, location coordinates are provided by degrees of latitude and longitude. Positive values of latitudes indicate those north of the equator, while negative values indicate those south of the equator[8]. Longitude measurements are relative to the zero meridian, with positive values extending east of the meridian and negative values extending west of the meridian. Supposing the coordinates of location A and B are $(latA, lonA)$ and $(latB, lonB)$, respectively, we have the formulae to calculate the distance between them on the Windows Azure Mobile Services website.

If the sum of two territories' radiuses is greater than the distance between them, the server judges it to be a collision and calls the battle method into play. The offender player's result probability can be calculated by his attack and his opponent's defense. If the probability is greater than a random number generated by the server, the offender will win; otherwise, his territory will be reduced. In a special case, when the distance is less than the winner's territory radius, the loser will lose the entire territory.

3.3.3 User authentication

Players in the game should be identified since this is intended to be a multi-user game. Windows Azure supports Facebook as an identity provider. With identification, the game can distinguish players. In addition, the notifications pushed to the players' devices can contain the players' personal information, such as their names.

We must first register as a Facebook developer. Since Windows Azure Mobile Services can be viewed as a website that provides a mobile application service, we create an application in the Facebook developer center with a website platform. After entering the web address of Azure Mobile Services, a Facebook application is created. Second, we supply the API (Application Program

Interface) Key and Facebook's Secret App to Windows Azure so that they can be linked together. Lastly, we modify our iOS application to show a log if the current player has not been identified. The identification information will then be passed to Windows Azure.

On the Windows Azure side, Mobile Services requires some detail information about the current user, such as name. Facebook provides an API for this usage. As long as an entity knows a user's access token, the user's social network information is available via the URL: https://graph.facebook.com/me?access_token=... (followed by the current token)[9].

This URL returns back a JSON (JavaScript Object Notation) object. JSON is a lightweight data format that is easy both for humans to read and write and for machines to parse and generate. In Windows Azure, the current user's name can be acquired by parsing the JSON object. The name will then be stored in the table storage. The next time the name is needed by a push notification message, it can be easily accessed.

4 Experimental Result

4.1 Test items

The following items were tested:

- We tested the pedometer first because it is a fundamental module in this game. A user carrying the device in his trouser pocket, which is where people commonly place their devices, walked a number of steps. We collected two step-number sets: one counted by the pedometer module and the other by the walker himself. The deviation of the pedometer module can be calculated with these two values.
- We next tested the communication between the iOS application and Windows Azure server. For convenience, a button that can increase the step number by 1,000 was added to the user interface of the iOS application. Thus, test participants did not actually have to walk a thousand steps in order to complete the experiment, which made the test procedure more efficient.
- Collision detection testing was the most important. The game is designed to be multi-player over the long term, which means that we cannot test the game in real situations. A large number of users and devices would be required and the test period would be too long, as it is almost impossible for a player to promote his knight level to a high value in just a few days and a participant would have to walk quite a long distance to reach a location that satisfies the test requirements. As a solution, we chose a method to simulate a true game scenario by modifying the data in the iOS application, similarly to the button used to virtually increase the number of steps.
- Finally, we tested the map view in the iOS application to verify that the correct territory information is displayed. Since the profile view that is designed to display knight-related information has not been

Figure 9. Push notification in iOS application

Figure 10. Map view in iOS application

implemented, it was not a test target. For the same reason, the server function that transfers daily game performance to knight experience was not tested.

4.2 Results

- As the former section described, we tested the pedometer module first. As Table 6 shows, the performance is acceptable when the device is kept in the user's trouser pocket. The average deviation rate was around 11.4%.

Table 6. Pedometer module performance

Test No.	Actual number	Detected number
1	1000	1093
2	1000	1128
3	2000	2241

- Collisions between territories were detected and processed by the game server. We also confirmed that the user authentication and push notification features worked efficiently. As shown in Figure 9, players received battle information via push notification when they obtained a territory, which also indicates that the player was identified correctly.
- In the map view, a player's territory location can be marked correctly with a red label. The corresponding territory information can also be displayed if the player taps that label, as shown in Figure 10.

Overall, we implemented the systematic function of the proposed walking incentive game. The iOS application can detect the player's step number, communicate with the game server, and display the territory in the map. Although the pedometer's performance seems to have low accuracy and use scenario, this does not overly affect the game's key function. Most of the other modules work without issue. The Windows Azure game server can detect and process territory collisions, push notifications to devices, and store game data.

5 Conclusion

We proposed a mobile game to encourage people to walk more and promote health. We designed the game rules and implemented our design on an iOS and Windows Azure platform. The systematic functions were tested and presented. Our work can be summarized as follows:

1. We proposed a healthcare-based mobile device game.
2. We discussed the game's rule design and balance.
3. We developed a multi-sensor application on iOS.
4. We replaced the conventional game server with the Windows Azure cloud service.

Although our original intent for implementation was simply to provide a prototype for experimental purposes, some notable issues remain, specifically:

- (a) As compared to other productive pedometers, the algorithm performance of our game's pedometer module needs to be improved.
- (b) Without game engine-rendered graphics, this game is more like a general application than a real game, although it has many game-related elements.
- (c) As the proposed system is under development, the App has several unfinished areas.

For example, the pedometer cannot detect steps when the application is running in background and the game server is not robust enough to cope with abnormal data.

Moreover, the proposed walking incentive game could be improved. It could be more social and more attractive by integrating closer with a social network. The software optimization could be significantly meaningful since the hardware resource is limited by the heat and battery life of mobile devices. Evaluation and analysis of the practical incentive effort of this game have yet to be completed.

Acknowledgement

The authors would like to express their sincere thanks to the anonymous reviews' careful review.

REFERENCES

- [1] M. Ma, F. Oliveira, et al *Serious Games Development and Applications: Third International Conference, SGDA 2012, Bremen*. Springer, LNCS 7528, 2012.
- [2] A. D. Cheok, A. Sreekumar, C. Lei, and L. N. Thang, "Capture the flag: Mixed-reality social gaming with smart phones," *IEEE Pervasive Computing*, vol. 5, no. 2, pp. 62–69, 2006.
- [3] Apple Inc, "About the iOS technologies," 2012, <https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/Introduction/Introduction.html>.
- [4] Microsoft Corp., "Introducing Windows Azure," <http://azure.microsoft.com/en-us/documentation/articles/fundamentals-introduction-to-azure/>.
- [5] Apple Inc, "Concepts in Objective-c programming," 2012, <https://developer.apple.com/library/ios/documentation/general/conceptual/CocoaEncyclopedia/CocoaEncyclopedia.pdf>.
- [6] Apple Inc, "iPhone Tech Specs," <http://support.apple.com/kb/sp587>.
- [7] Google Inc, "Introduction to the GoogleMaps SDK for iOS," 2012, <https://developers.google.com/maps/documentation/ios/intro?hl=ja>.
- [8] Geography All the Way, "Online geography resources. latitude and longitude," http://www.geographyalltheway.com/ks3_geography/maps_atlases/longitude_latitude.htm.
- [9] Facebook Inc, "Facebook APIs, graph API, user object," <https://developers.facebook.com/docs/graph-api/reference/v2.1/user>.