International Journal of Smart Home
Vol. 7, No. 3, May, 2013

An Indoor Golf Simulator for Continuous Golf Games

KeeHyun Park and SeungHyeon Lim

Department of Computer Engineering, Keimyung University, South Korea
khp@kmu.ac.kr, burningwing@kmu.ac.kr

Abstract

An indoor golf simulator is an augmented game simulator. Because an indoor golf simulator can be easily installed in indoor golf centers, fitness centers, offices, and residential houses, it has become one of the hottest products on the market, allowing golfers to enjoy their games indoors. However, the existing indoor golf simulators require that a player finish his game without a break. This paper proposes an indoor golf simulator that allows a player to pause a game at any time and resume it at any place. The proposed simulator applies data synchronization techniques in ubiquitous environments.

The simulator consists of PC consoles, a proxy server, and a player data server. The PC console client uses a XML-based communication protocol to send information about the game to the player data server via the proxy server. The communication protocol supports DS (Data Synchronization) operations between the PC console client and the player data server. Located between the PC console client and the player data server, the proxy server receives game information from the PC console client and sends the information to the player data server, and vice versa. When the player data server authenticates the player, the proxy server establishes a new communication session and manages the session until the game is either terminated or suspended. The player data server authenticates the player and keeps his or her latest game information for later use (e.g., to resume the game). Each component of the simulator is designed and explained in detail. By applying DS protocols, the proposed simulator allows a player to pause his or her game at any time and resume it at any place.

Keywords: *indoor golf simulator, continuous golf game, data synchronization*

1. Introduction

An indoor golf simulator is an augmented game simulator. It consists of a screen, a sensor system, a projector, and a PC [1-3]. Because an indoor golf simulator can be easily installed in indoor golf centers, fitness centers, offices, and residential houses, it has become one of the hottest products on the market, allowing golfers to enjoy their games indoors. However, the existing indoor golf simulators require that a player finish his game without a break. This paper proposes an indoor golf simulator that allows a player to pause a game at any time and resume it at any place. The proposed simulator applies data synchronization techniques in ubiquitous environments [4].

The simulator consists of a PC console, a proxy server, and a player data server. The PC console client is installed in a PC console to provide an indoor golf game and transmit information about the game to the proxy server. Located between the PC console client and the player data server, the proxy server manages a session. The player data server authenticates the player and keeps his or her latest game information for later use (*e.g.*, to resume the game).

The rest of the paper is organized as follows: Section 2 describes the related studies. Section 3 explains in detail a communication model that supports continuous indoor golf games. Finally, Section 4 draws conclusions and discusses future directions of research.

2. Background

2.1. Indoor Golf Simulator

An indoor golf simulator is an augmented game simulator, consisting of a screen, a sensor system, a projector, and a PC. Using the projector and PC console, the simulator displays 2D or 3D images of real golf courses on the screen. The sensor system analyzes the player's strokes, measuring movements, such as ball speed, direction, angle, distance, and so on. Based on the information transmitted from the sensor system, the PC console generates the images of the ball's movements on the screen.

2.2. Data Synchronization

Data synchronization (DS) is the technology used to keep all data replicas up-to-date and consistent in both distributed and networked environments. Because the device is mobile, users can access their data at any time in any place. To keep the data up-to-date and consistent, data changes must be communicated to each device. Currently, there are several DS protocols for mobile environments, such as Palm's HotSync [5], Microsoft's ActiveSync [6], and OMA DS [7]. These protocols detect all updates and apply them to each device. To accomplish this, they copy the data from mobile devices to a DS server and keep the data up-to-date and consistent by providing communication between the server and mobile devices.

OMA DS is an open industry-standard data synchronization protocol for mobile devices. All mobile devices synchronize with a DS server by using the SYNCML-based OMA DS protocol [8-10]. Figure 1 illustrates the flow of the OMA DS protocol, which consists of three processing procedures (session initialization, data synchronization, and session release) and six packages.

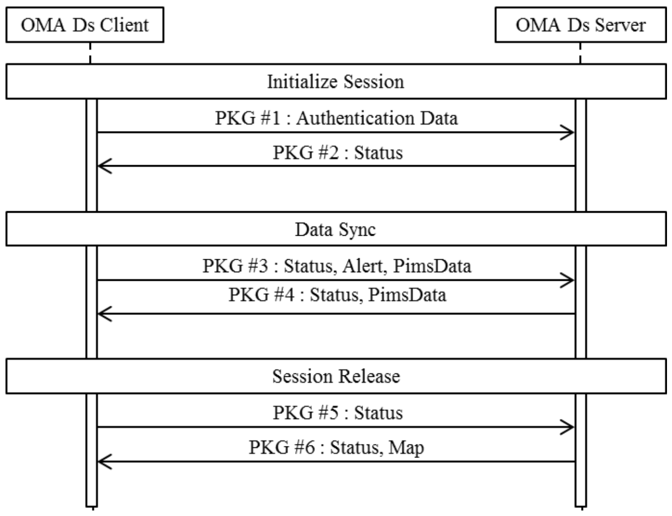


Figure 1. OMA DS Protocol

3. Communication Model

3.1. Structure of Communication Model

The communication model proposed for a continuous golf simulator game consists of PC consoles, a proxy server, a player data server and a communication protocol based in XML [11], as shown in Figure 2.

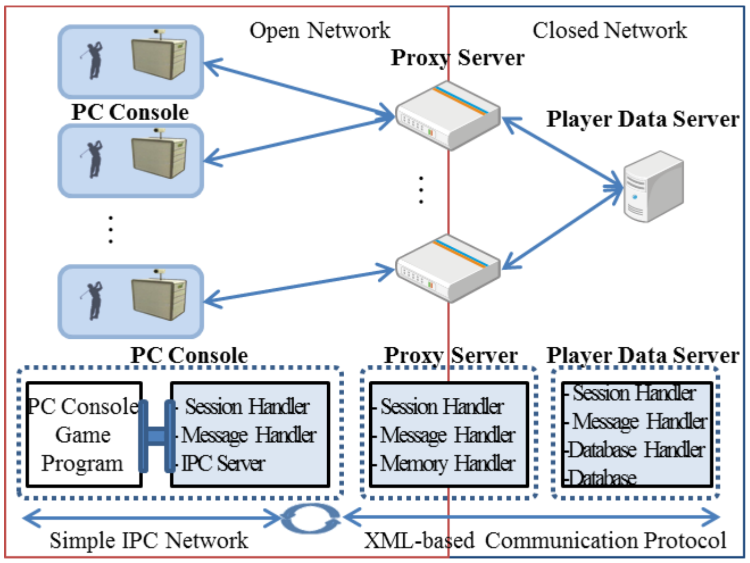


Figure 2. The Structure of the Communication Model

Most existing indoor golf game simulators have PC consoles only. They do not access servers that support PC consoles. In these game simulators, a PC console is the only device responsible for a golf simulator game involving a player. However, in the golf game simulator proposed in this paper, in addition to PC consoles, a proxy server and a player data server are used to support the continuity the indoor golf game. When a player logs into the golf game simulator, a PC console client installed in a PC console functions as a client program that communicates with a proxy server. Located between the PC console client and the player data server, the proxy server is responsible for data communication and session management. All data related to the game in which the player has been involved are stored in the player data server for later use. In order to resume the golf game, game information can be downloaded to the PC console client, which is not necessarily the same PC console as the one that the player used to pause the game.

3.2. PC Console Client

Figure 3 illustrates the structure of a PC console. It consists of a PC console program and a PC console client.

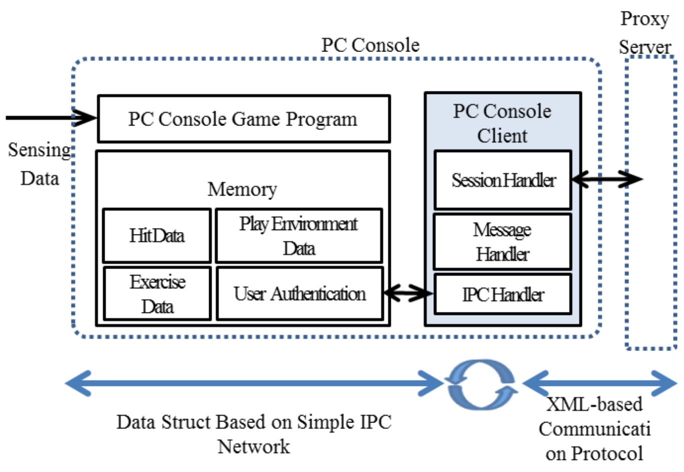


Figure 3. The Structure of a PC Console

3.2.1. PC console game program

The main responsibilities of the PC console program are the same as those of a traditional golf game simulator program. The PC console program displays images of the golf course the player chooses on the screen, receives data about the movement of the ball the player hits from the sensor system, and displays the movement of the ball on the same screen. In addition to these main responsibilities, the PC console game program stores the game information for transmission to the player data server via the proxy server at the appropriate times described in Table 1.

Table 1. Times to store game information

Times to store	Game Information to be stored
At the beginning of a game	Field and hole number, Player
At the beginning of a hole	Player sequence number, Hole number
At the end of a hit	Player sequence number, Ball position
At the end of a hole	Number of remaining holes, Player
At the end of a game	Player

3.2.2. PC console client

The PC console client uses a XML-based communication protocol to send information about the game to the player data server via the proxy server. The communication protocol supports DS operations between the PC console client and the player data server. The DS operations are explained below. A PC console client consists of an IPC handler, a session handler, and a message handler:

- IPC handler: Using the IPC method, the IPC handler communicates with the PC console game program installed in the same PC console.

- Session handler: It manages communication sessions between the PC console client and the related proxy server.

- Message handler: It generates, accesses, and analyzes XML-based messages.

3.2.3. Proxy server

The structure of a proxy server is shown in Figure 4. Located between the PC console client and the player data server, the proxy server receives game information from the PC console client and sends the information to the player data server, and vice versa. When the player data server authenticates the player, the proxy server establishes a new communication session and manages the session until the game is either terminated or suspended. The proxy server consists of a memory handler, a session handler, and a message handler:

- Memory handler: It keeps a list of sessions authenticated by the player data server.

- Session handler: It manages communication sessions between the PC console client and the related player data server.

- Message handler: It encrypts and decrypts XML-based messages.

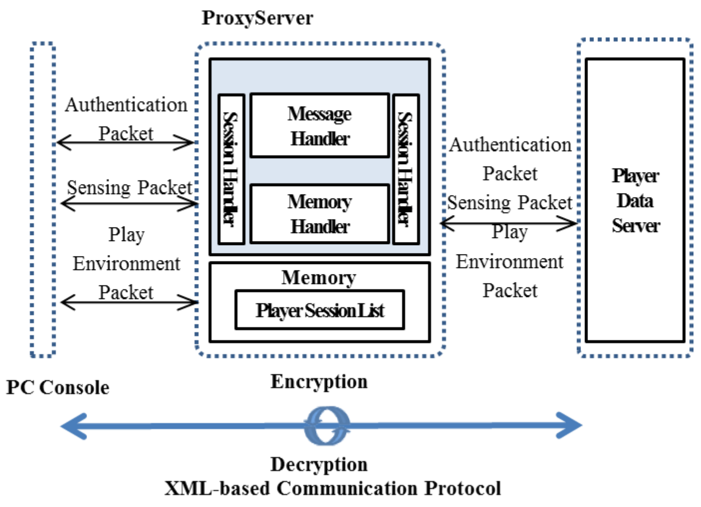


Figure 4. The Structure of a Proxy Server

3.2.3. Player data server

A player data server is shown in Figure 5. The player data server authenticates the player (or PC console client). It also manages the database, which stores the game information. The PC console client consists of a database handler, a session handler, and a message handler:

- Database handler: It stores the game information in the database. It also accesses the appropriate game information and sends it to the related proxy server when it receives a request to resume a game.

- Session handler: It manages communication sessions between the player data server and the related proxy server.

- Message handler: It analyzes the XML-based messages received from the related proxy server and generates XML-based response messages.

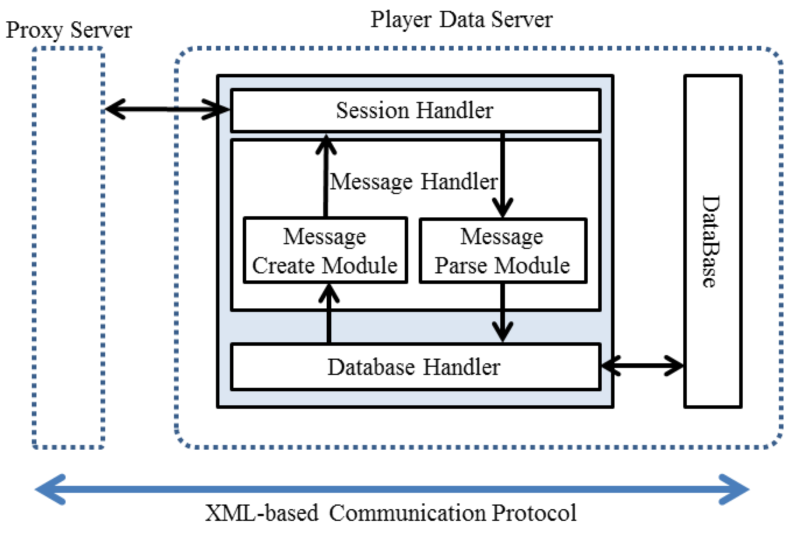


Figure 5. The structure of a player data server

Functions of the player data server are represented in the Table 2. The player data server can restore the recent game environment data at any time because the server stores repeatedly data transmitted just after the moments of the beginning of a new game, the beginning of a new hall, and swing.

Table 2. Functions of Player Data Server

Function	Description
- NewGameCreate	Store the selected fields, attributes, and player ID for a new game
- HoleStart	Store the current hole number and player ID
- ShotAfter	Store player ID and data about the movement of the ball just after the player takes a swing
- GameEnd	Store game environment data in the database
- Resume	Tell who is the next player when a game is resumed
- Suspend	Store the current status when a game is paused

3.2.3. Communication protocol

Figure 6 illustrates the flow of the communication protocol for the system [13]. The protocol performs data synchronization operations between the PC console programs and the player data server. The communication protocol makes it possible for game information to be consistent between the PC console programs and the player data server by reflecting any changes in game information stored in the PC console program. The changes are transmitted to the player data server via the proxy server whenever game information is stored or updated in the PC console.

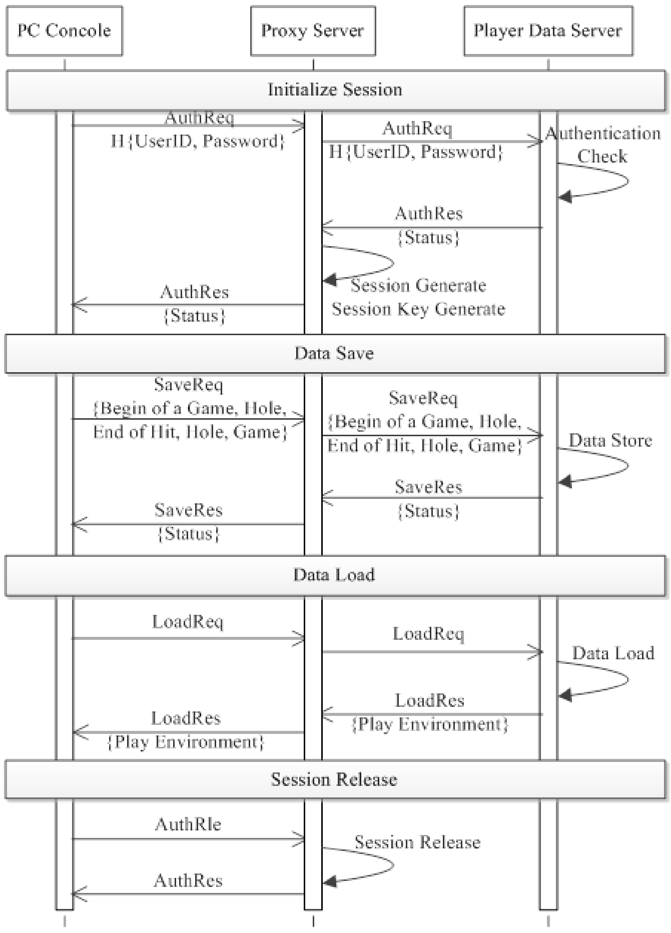
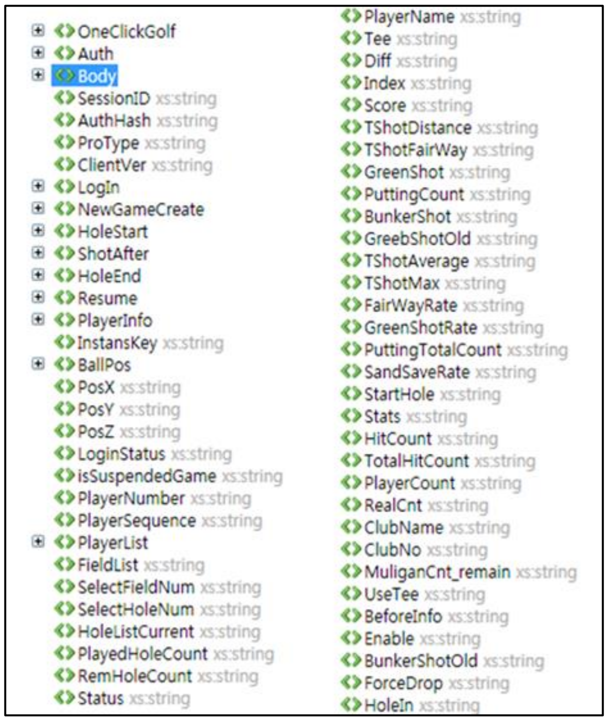


Figure 6. Flow of the Communication Protocol

The communication protocol consists of four phases: session initialization, data load, data save, and session release. The session initialization phase, the data load phase, and the session release phase are executed only once, whereas the data save phase can be executed more than once according to the times that game information is stored, as shown in Table 1. The data load phase is executed when a game resumption is requested.



```
OneClickGolf                          PlayerName xs:string
Auth                                  Tee xs:string
Body                                  Diff xs:string
SessionID xs:string                   Index xs:string
AuthHash xs:string                    Score xs:string
ProType xs:string                     TShotDistance xs:string
ClientVer xs:string                   TShotFairWay xs:string
LogIn                                 GreenShot xs:string
NewGameCreate                         PuttingCount xs:string
HoleStart                             BunkerShot xs:string
ShotAfter                             GreebShotOld xs:string
HoleEnd                               TShotAverage xs:string
Resume                                TShotMax xs:string
PlayerInfo                            FairWayRate xs:string
InstansKey xs:string                  GreenShotRate xs:string
BallPos                               PuttingTotalCount xs:string
PosX xs:string                        SandSaveRate xs:string
PosY xs:string                        StartHole xs:string
PosZ xs:string                        Stats xs:string
LoginStatus xs:string                 HitCount xs:string
isSuspendedGame xs:string             TotalHitCount xs:string
PlayerNumber xs:string                PlayerCount xs:string
PlayerSequence xs:string              RealCnt xs:string
PlayerList                            ClubName xs:string
FieldList xs:string                   ClubNo xs:string
SelectFieldNum xs:string              MuliganCnt_remain xs:string
SelectHoleNum xs:string               UseTee xs:string
HoleListCurrent xs:string             BeforeInfo xs:string
PlayedHoleCount xs:string             Enable xs:string
RemHoleCount xs:string                BunkerShotOld xs:string
Status xs:string                      ForceDrop xs:string
                                      HoleIn xs:string
```

Figure 7. Schema of the Communication Protocol

The communication protocol is represented in XML. Figure 7 represents the XML schema of the communication protocol, which consists of two elements - Auth and Body. The Auth element consists of an authentication data, a ProtoType data, and the general data for the PC console client. The ProtoType data represents the usage of the XML documents. The contents of the Body element vary according to the ProtoType data stored in the Auth element. In general, a Status data, which represents the processing status of a message, and game environment data are stored in the Body element.

4. Conclusion

In this paper, an indoor golf simulator designed to allow continuous golf games is proposed. The simulator consists of PC consoles, a proxy server, and a player data server. Each component of the simulator is designed and explained in detail. By applying DS protocols, the proposed simulator allows a player to pause his or her game at any time and resume it at any place. Because the XML-based DS protocol is used, protocol upgrades and updates can be performed easily. Locating the proxy server between the PC console and the player data server facilitates player authentication.

Acknowledgements

This work (Grants No. 0002985) was supported by Business for Academic-industrial Cooperative establishments funded Korea Small and Medium Business Administration in 2012.

References

[1] S. -I. Lee, "The effect of motivation of virtual reality sports and service factors on leisure satisfaction", Master's Thesis, Sejong University, **(2011)**.
[2] S. -H. Ahn, "Development of a simulator for Interactive 3D Golf Game", Master's Thesis , Hallym University, **(2007)**.
[3] B. -m. Moon, "The Study of Screen Golf's fun factors on exercise immersion experience, participating satisfaction and exercise continuation behavior", Master's Thesis, Silla University, **(2010).**
[4] M. R. Lee and T. T. Chen, "Trends in Ubiquitous Multimedia Computing", International Journal of Multimedia and Ubiquitous Engineering, vol. 4, no. 2, **(2009)**, pp. 115-124.
[5] HotSync, http://www.accessdevnet.com/docs/conduit/win.
[6] D. Boling, "Programming Microsoft Windows cs.net", Microsoft Press, **(2004)**.
[7] OMA DS Protocol Specification Version 1.2, http://www.openmobilealliance.org.
[8] U. Hansmann, R. Mettala, A. Purakayastha and P. Thompson, "SyncML Schronizing and Managing Your Mobile Data", Prentice Hall Pub. Co., **(2003)**, pp. 21-24.
[9] J. -G. Pak and K. Park, "Construction and Validation of a Data Synchronization Server upporting OMA DS Standards", Journal of the Korea society of computer and information, vol. 16, no. 5, **(2011)**, pp. 79-91.
[10] K. Park and J. -G. Pak, "Efficient Transmission Method for Mobile Data Synchronization Based on Data Characteristics", Lecture Notes in Electrical Engineering, vol. 120, **(2011)**, pp. 253-266.
[11] Y. Song, K. Choo and S. Lee, "Design of Index Schema based on Bit-Streams for XML Documents", International Journal of Software Engineering and Its Applications. vol. 6, no. 4, **(2012)**, pp. 131-136.
[12] K. Park and J. -G. Pak, "Construction of a Medication Reminder Synchronization System Based on Data Synchronization", International Journal of Bio-Science and Bio-Technology. vol. 4, no. 4, **(2012)**, pp. 1-10.
[13] K. Park and S. H. Lim, "A Communication Model to Support Continuous Indoor Golf Simulator Games", Submitted to SoftTech2013 Conf. for publication, **(2013)**.

Authors

KeeHyun Park received his B.Sc. and M.Sc. degrees in Computer Science from Kyungbook National University, Korea, and from KAIST, Korea, in 1979 and 1981, respectively, and his Ph.D. degree in Computer Science from Vanderbilt University, USA, 1990. He has been a professor of Computer Science and Engineering Department at Keimyung University, Korea since March 1981. His research interests include Mobile/Network Communication System, Embedded System and Parallel Processing System.

SeungHyeon Lim received his B.Sc. Computer Engineering from Keimyung University, Korea, in 2012. His research interests include Mobile Device Management and Data Synchronization.