A NEW CONCEPT FOR REAL-TIME WEB GAMES
Developing Highly Real-Time Web Games

Yoshihiro Kawano[1], Masahiro Miyata[2], Dai Hanawa[3] and Tatsuhiro Yonekura[4]
[1]*Intec NetCore, Inc.,Tokyo, Japan*
[2]*Department of Computer and Information Science, Ibaraki University, Ibaraki, Japan*
[3]*Department of Computer and Information Science, Faculty of Science and Technology, Seikei University, Tokyo, Japan*
[4]*Department of Computer and Information Science, Ibaraki University, Ibaraki, Japan*

Keywords: Web game, Real-time, Ajax, Information extraction layer, Dead reckoning, Allocated Topographical Zone.

Abstract: Online games have rapidly gained popularity as network speed and computer performance has improved. Network latency remains a significant problem, though, in applications needing high interactivity, such as action games and real-time sports games. Such applications are called a Distributed Virtual Environment (DVE). In general, a server-client architecture is used for a DVE. In this paper, we focus on the current Web system, the most widely used infrastructure of this type, and propose a strategy for designing real-time Web games. To date, no such design strategies have been explicitly established. In an earlier paper, we introduced two techniques, called Dead reckoning and Allocated Topographical Zone. Here, we explain how these can be used in real-time Web games to overcome the limitations of HTTP communication. As an example of such application, we have implemented a Web-based real-time avatar operation game. This implementation confirmed that our concept can be extended to various types of real-time Web application.

1 INTRODUCTION

Online games are increasingly popular due to improved network speed and computer performance. Network latency is a significant problem, however, in applications requiring high interactivity, such as action games and real-time sports games (Takemura, 1999). Such applications are called a Distributed Virtual Environment (DVE). In general, a server-client architecture is used for a DVE (Takemura, 1999).

Examples of highly real-time online games are Counter-Strike (Valve Corporation, 2007), which is a first-person shooter (FPS) game, and Age of Empires III (Microsoft Corporation, 2007), a real-time strategy (RTS) game. Another example is SHIN-SANGOKU MUSOU BB (KOEI Co., Ltd., 2007), which is a highly real-time multiplayer action game that is only open to users of a particular Internet service provider (ISP). To play such online games, though, the user has to install dedicated client software for each game. Moreover, if the software uses a particular port-number, it might not be possible to transmit the necessary data past the user's firewall.

In this paper, we focus on use of the Web as the infrastructure to support online games because it is the most widely used. When a game is actualized on the Web, users can easily play it through a browser, and this tends to lead to growth in the number of users. On the other hand, if HTTP is used to transmit the data, the transmission will not be limited by a firewall. Such a technique can also be applied for educational applications.

The Web game Dinoparc (Motion-Twin, 2005) is an example of a multiplayer online game available on the Web, although no highly real-time Web games of this type have been reported. On the other hand, a JAVA-based MMORPG (massively multi-player online role playing game) called RuneScape is available on the Web (Jagex Ltd., 2007), but this game does not allow highly real-time interaction. A Shockwave-based MMOFPS real-time Web game called Tank Ball2 is also available (Maid Marian Entertainment, 2004), but the consistency of collision detection between tank and ball is questionable even though the collision detection is the most critical factor in the game. Such a lack of consistency in real-time games means that shared objects cannot be jointly controlled by all users

Kawano Y., Miyata M., Hanawa D. and Yonekura T. (2008).
A NEW CONCEPT FOR REAL-TIME WEB GAMES - Developing Highly Real-Time Web Games.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 171-177
DOI: 10.5220/0001525701710177

because of temporal differences between the local and the remote avatars (virtual players) in terms of their status in reference to each other (Hashimoto, Sheridan and Noyes, 1986).

Thus, we have studied ways to improve consistency for all states, including those of the avatars and shared objects in a real-time Web game where a shared field is equally accessible to and controllable by each terminal, without sacrificing throughput. We adopted the virtual ball game (VBG) as an example of a real-time Web game. To construct a VBG, in which the objects shared by the players have the physical attributes of location, orientation and velocity, a significant issue is how the ownership (the decision as to which terminal owns the key to control an object) and the attributes for all avatars and objects can be kept consistent among terminals. Based on our work, in this paper we propose a new concept for designing real-time Web games.

2 REAL TIME WEB GAME REQUIREMENTS

Three requirements need to be met to achieve a VBG type real-time Web game:

(1) Consistency in the attributes of avatars
(2) Throughput of the operations of users
(3) Speed and accuracy in the collision detection for a shared object

HTTP data transmission using the Web has to satisfy these three requirements, but HTTP transmission for real-time games suffers from transmission lag, bandwidth limitation, and jitter. Our goal has been to overcome these critical issues to enable the playing of real-time games on the Web.

To meet requirements (1) and (2), the game logic part should be separate from the data transmission part. For this purpose, we can use Ajax (Asynchronous JavaScript + XML) (OpenAjax Alliance, 2007). Ajax offers several advantages:

- Data can be transmitted asynchronously
- Game logic can be executed on the local client side
- Parts can be changed only when necessary

When Ajax is used to support a real-time Web game, each client executes a modelling-view-control (MVC) loop and organizes a virtual world by himself; thus, the server load can be decentralized. Therefore, in this paper, our objective is Web games that use Ajax. This concept permits some inconsistency between the information represented on each terminal because the network can be considered a virtual peer-to-peer (P2P) model. Synchronization between terminals is not easily achieved, however, because such a network has no server to provide synchronization management, as does a server-client network. Therefore, it is necessary to maintain consistency of the presented information through Dead reckoning (DR) on each terminal.

To meet requirement (3), we used the Allocated Topographical Zone technique (AtoZ) (Kawano and Yonekura, 2006). When AtoZ is used in a real-time P2P VBG, each peer can clearly determine the ownership of shared objects without synchronization between peers regarding the ownership.

3 SYSTEM CONCEPT

In this section, we propose a new concept for designing real-time Web games.

3.1 Information Extraction Layer

First, we propose a new hierarchical structure model concept (Figure 1). The information extraction layer (IEL) is a new layer in the model and it extracts value from information that could be useful for the application or the user. For example, a Google search engine extracts search results from Web sites worldwide, and an access analysis tool extracts access statistics for a Web site from access logs. In this paper, DR to predict a remote avatar's attributes and AtoZ to determine the ownership of shared objects are IEL functions. Therefore, the IEL mediates between the network and the application (contents).

3.2 System Architecture

Figure 2 shows the system architecture designed to satisfy the requirements given in Section 2, and reveals how the system architecture lays over the hierarchical structure model from Figure 1.

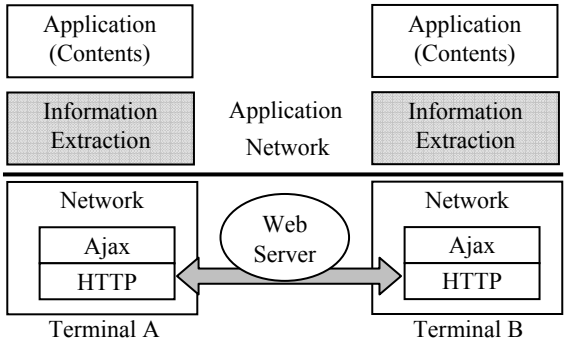

Figure 1: Hierarchical structure model.

In Figure 2, each terminal has five components: GUI Handler, Modeler, Viewer, Virtual Space Resolver (VSR), and Network Handler. There is an application layer MVC loop on each client side. The interval of each MVC loop is asynchronous with that of the network transmission with Ajax (Figure 3). In Figure 3, the broken circle and arrow indicate transmission. As shown, a local terminal acquires information from a remote terminal within an MVC loop interval based on the data received in the network transmission interval.

In this paper, we focus on the roles of the IEL and the VSR implemented on the IEL. The IEL mediates between the application (the GUI Handler and the Modeler) and the Network Handler, and manages the virtual space with respect to the requirements given in Section 2.

3.3 System Components

Here, we describe the role of the Web server and each component making up the MVC loop.

3.3.1 Web Server

The Web server acts as part of the medium connecting terminals. The data used to acquire a virtual space (for example, the position of an avatar) is transmitted using the JavaScript Object Notation (JSON) (Internet Engineering Task Force, 2006) format instead of XML format to reduce the data size and parsing time (Ward, 2007).

The Web server receives the data for an avatar on each terminal according to the HTTP request timing. At that time, the server combines the latest data of each terminal like a chain, and sends it as an HTTP response.

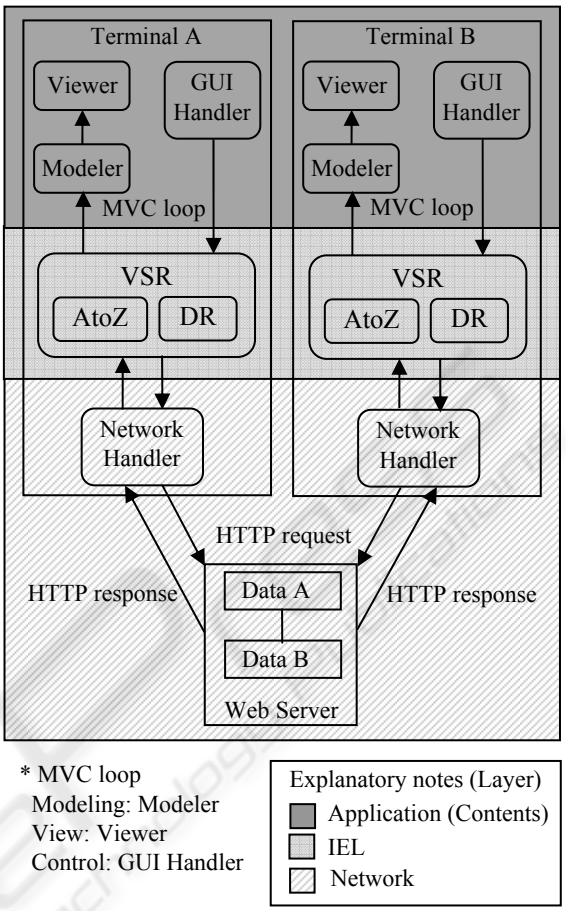

Figure 2: System architecture.

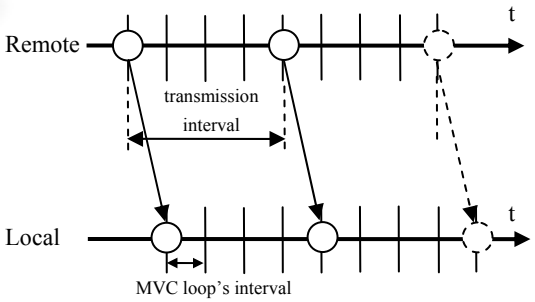

Figure 3: Time-chart of MVC loop and network transmission.

3.3.2 GUI Handler

The GUI Handler sends the user input to the VSR, where it is computed and then provided to the Modeler.

3.3.3 Modeler

The Modeler receives information from the VSR, and then acquires a virtual space based on the information and provides the virtual space to the Viewer.

3.3.4 Viewer

The Viewer presents the game information provided by the Modeler to a user. For this part, we use a Web browser as the Viewer.

3.3.5 Virtual Space Resolver

The VSR includes DR and AtoZ, and resolves issues regarding the virtual space (avatars and shared objects) described in Section 2. The VSR receives data regarding the user's operation, and provides the appropriate information to the Modeler. The VRS also receives the latest data concerning remote avatars from the Network Handler independently of the above transmission. The VSR is a core component in this system.

There is a time interval before the data are received from the Network Handler. This time interval can negatively affect the smoothness of the game progression. To avoid this, the data of remote avatars are extrapolated by DR (Singhal, 1996) based on data with a time stamp. DR is also used in networked computer games and simulations to reduce lags caused by network latency and bandwidth issues.

In this system architecture, synchronization regarding the ownership of shared objects between terminals is not easily achieved because the architecture has no server to provide synchronization management. To overcome this problem, which is necessary to meet requirement (3) of Section 2, we use AtoZ as a function of the VSR.

Thus, the objective of the VSR is to provide appropriate information to the Modeler with no need for a game developer to consider the effects of network latency. Accordingly, game developers can realize Real-time Web games by simply creating content like a stand alone program.

3.3.6 Network Handler

The Network Handler uses Ajax to control the transmission part to the Web server.

3.4 Advantages of the Design

The proposed design offers three main advantages:

(1) Distributed game computing
(2) Need to develop game content only
(3) Applicability of the mash-up technique

Regarding (1), the construction of a virtual space, which is computed on the server in current Web systems, can be done on each terminal and the server load is reduced. Real-time Web games can therefore be realized. Regarding (2), the VSR and the Network Handler are made to a single component and provided as an API. As a result, a Web game developer has to create only the game content. Regarding (3), using the mash-up technique and combining the APIs of other site make it possible to provide new services that provide additional value.

4 SYSTEM DESIGN

In this section, we describe the specific system design.

4.1 System Design Strategy

The system design goal was to implement our technology in the DVE for real-time Web games. We have described the DR protocol (Hanawa and Yonekura, 2007) and AtoZ (Kawano and Yonekura, 2006) elsewhere, and in this paper we examine the implementation of these techniques.

4.2 Dead Reckoning Protocol

The DR protocol provides certain advantages. That is, the DR protocol extrapolates the remote data during transmission interval. Therefore, even if the transmission interval is limited like HTTP, information can be presented to user in real-time without the effects of network latency. In this section, we explain how the protocol can be applied to a real-time web game. Figure 4 describes the details of the DR component from Figure 2. The hierarchical structure (Application and the IEL) is indicated by solid lines, and the three components (the VSR, the GUI Handler and the Modeler) are indicated by broken lines.

In Figure 4, $A(t_i)$ and $A'(t_i)$ indicate the position of avatar A at time t_i and the differences of avatar A at time t_i, respectively, while $\underline{A}(t_{i+1})$ indicates the predicted position of avatar A at time t_{i+1}.

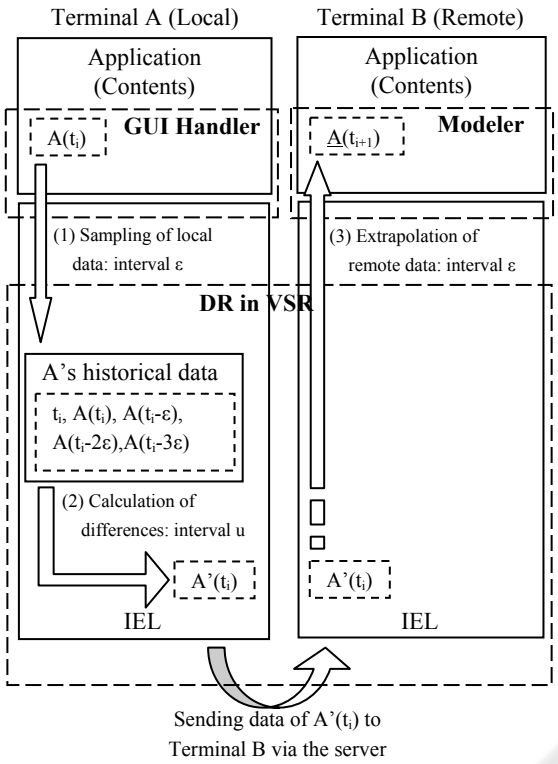

Figure 4: The dead reckoning protocol in our system.

As shown in Figure 4, the DR protocol in our system consists of three steps:

(1) Sampling of local data
First, the local terminal (Terminal A in Figure 4) records data for the local avatar during interval ε, which is a very small interval time. The data is used to update the local historical data, where the sampling data format is as shown in Figure 5a.

(2) Calculation of the differences
Second, the local terminal calculates the differences based on the historical data, and these calculated differences are transmitted during the interval *u* to the remote terminal (Terminal B in Figure 4). The transmitted data format is shown in Figure 5b.

(3) Extrapolation of the remote data
Third, the remote terminal extrapolates the data for the local avatar during interval ε.

In the above, local sampling interval ε is set to less than the update interval *u*. In addition, the DR protocol is utilized as a function of the VSR.

```
{ "name" : "name of the avatar (= ID)",
  "Px ": x component of the position at time t,
  "Py" : y component of the position at time t,
  "Pz" : z component of the position at time t,
  "t"  : current time }
```

a. Sampling data format.

```
{ "name" : "name of the avatar (= ID)",
  "Px" : x component of the position at time t,
  "Py" : y component of the position at time t,
  "Pz" : z component of the position at time t,
  "Vx" : x component of the velocity at time t,
  "Vy" : y component of the velocity at time t,
  "Vz" : z component of the velocity at time t,
  "Ax" : x component of the acceleration at time t,
  "Ay" : y component of the acceleration at time t,
  "Az" : z component of the acceleration at time t,
  "t"  : current time }
```

b. Transmitted data format.

Figure 5: JSON format used with dead reckoning protocol.

4.3 AtoZ (Allocated Topographical Zone)

The AtoZ distributed processing protocol that we can use for a P2P-type VBG has been described elsewhere (Kawano and Yonekura, 2006). Here, we focus on how using AtoZ enables us to introduce a VBG on the Web. In the MVC loop of Figure 2, the AtoZ component is included as a function of the VSR in the same way as the DR protocol explained in the previous section.

4.3.1 A P2P-Type Virtual Ball-Game

In a P2P-type VBG, each avatar and shared object has physical attributes (location, orientation and velocity). A significant issue is how ownership (determination of which peer owns the key to control a shared object) and the attributes for all avatars and shared objects can be kept consistent among peers. In addition, each player, or peer, can manipulate exclusively the avatar dedicated to that peer (no other player can manipulate that avatar). Each shared objects can be continuously under the control of only a certain avatar that is manipulated by a certain player. In other words, only one avatar exclusively owns the shared object at a given time. The decision-making regarding ownership is constantly computed, and ownership is dynamically transferred from one avatar to another.

4.3.2 Introduction of AtoZ

For use under such conditions, we developed the following distributed cooperative protocols for application among peers regarding ownership of a shared object.

First, let's consider that each avatar has a priority field in which the ownership is given to that avatar while the shared object remains within the field. The priority field is defined as a set of spatial points in order to decide which avatar can gain access privilege to the shared object, and this must be uniquely decided among the avatars. Hence, the priority field is decided dynamically, considering the relationship between the associated avatars in terms of their position, velocity, acceleration and orientation. In this paper, the priority field is referred to as the AtoZ field formulated as

$$AtoZ(p_i) = \{\boldsymbol{x} \mid \boldsymbol{x} \in Z, Access(\boldsymbol{x}, p_i) = \min_{j \in N}(Access(\boldsymbol{x}, p_j))\} \quad (1)$$

where p_i denotes the i'th avatar (i = 1,2,…,N).

In the formula above, AtoZ(p_i) is the AtoZ field of avatar p_i, **x** and Z respectively denote a spatial point and the whole space in the DVE, and $Access(\mathbf{x}, p_i)$ denotes the estimated elapsed time needed for avatar p_i to reach out to x. Thus, AtoZ(p_i) indicates the set of points that avatar p_i can reach out to faster than any other avatar. This is illustrated in Figure 6.

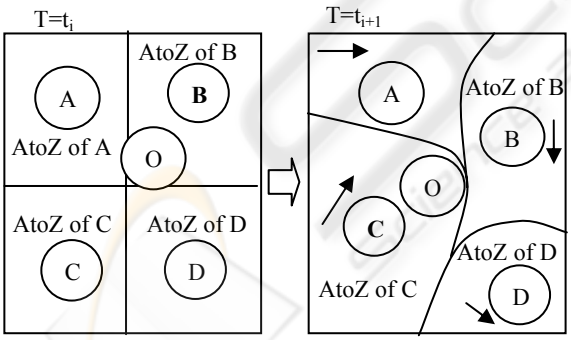

Figure 6: Overview of AtoZ.

If we assume that each peer uniquely determines the attributes of each avatar, each peer also commonly computes the AtoZ values for all avatars because they use the same algorithm to determine the AtoZ value (i.e., priority field) with the same calculation accuracy. Thus, every peer may, though acting independently, select the same avatar as the one having ownership of a shared object because all peers follow the same decision-making rule regarding ownership. This satisfies the requirement for ownership consistency.

4.4 Example of the Application

We implemented an avatar operation game as a prototype system. In this system, a user who logs into the web site can operate his own avatar, and the motion of this and other avatars is displayed through the browser. Figure 7 is an overview of the system, and Figure 8 is an application screenshot. In this application, we set the MVC loop interval to 100 ms and the network transmission interval to 300 ms. The URL of this Web site is http://yard.cis.ibaraki.ac.jp/webgame/login.jsp

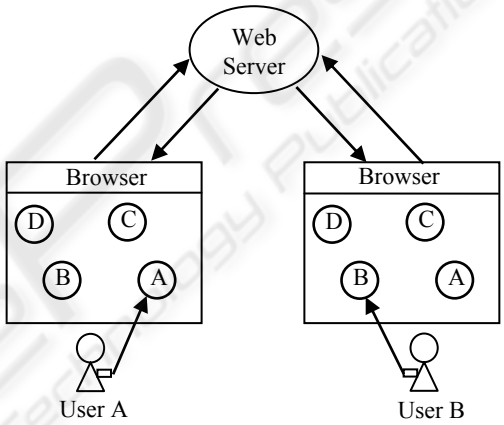

Figure 7: System overview of an avatar operation game.

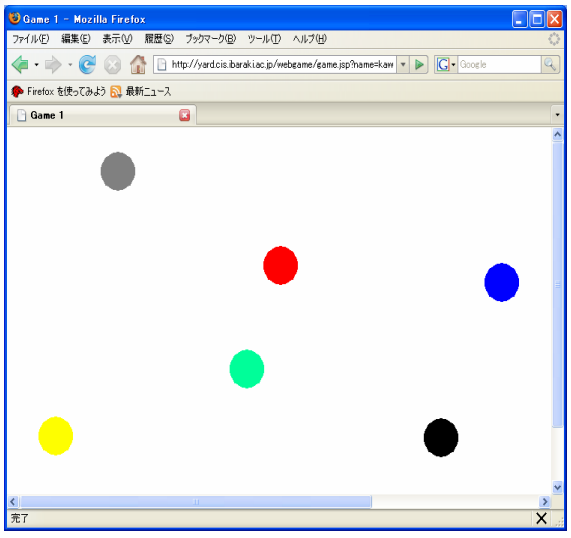

Figure 8: Application screenshot.

5 CONCLUSIONS

We have described a new approach to designing real-time Web games where we use DR and AtoZ to overcome the limitations of HTTP communication. As an example of this sort of application, we have implemented a Web-based real-time avatar operation game. This has demonstrated that our concept can be extended to various types of real-time Web application.

Our future work will involve the enhancement of Web games based on our concept and further evaluation of the concept's validity throughout games. We will also consider other types of application in addition to Web games and systematize our concept.

REFERENCES

Takemura, H., 1999. *Journal of the Virtual Reality Society of Japan,* vol.4, no.1, pp.59-63, (in Japanese).

Hashimoto, T., Sheridan, T.B., Noyes, M.V., 1986. "Effects of predicted information in teleoperation with time delay", *Japan Ergonomics Society*, vol.22, no.2, pp.91-92, (in Japanese).

Valve Corporation, 2007. Counter-Strike, http://www.steamgames.com/v/index.php?area=game&AppId=240.

Microsoft Corporation, 2007. Age of Empires III, http://www.ageofempires3.com/.

KOEI Co., Ltd., 2007. SHIN-SANGOKU MUSOU, http://www.musou-bb.jp/.

Motion-Twin, 2005. Dinoparc, http://www.dinoparc.com/

OpenAjax Alliance, 2007. http://www.openajax.org/index.html.

The Internet Engineering Task Force, 2006, RFC4627: The application/json Media Type for JavaScript Object Notation (JSON), http://www.ietf.org/rfc/rfc4627.txt.

Singhal, S.K., 1996. Effective remote modeling in large-scale distributed simulation and visualization environments, *Ph.D. Dissertation, Department of Computer Science,* Stanford University, Palo Alto.

Kawano, Y., Yonekura, T., 2006. Count Down Protocol: Asynchronous Consistent Protocol in P2P Virtual Ball Game for Multi-Player, *in Proc. ACM NetGames2006*.

Hanawa, D., Yonekura, T., 2007. Improvement on the accuracy of the polynomial form extrapolation model in distributed virtual environment, Springer-Verlag, Visual Comput, 23: pp.369-379.

Maid Marian Entertainment, 2004. Tank Ball2, http://maidmarian.com/Tank.htm.

Jagex Ltd., 2007. RuneScape, http://www.runescape.com/

James Ward, 2007. Census - RIA Data Loading Benchmarks, http://www.jamesward.org/census/.