A computer implemented method, a data processing system, and computer program product are provided for monitoring user game-playing behavior and reporting game play data in a gaming environment. A software agent is provided for monitoring gaming characteristics of the user responsive to a launch of a game by a user. The software agent also identifies a game state based on the gaming characteristics. The software agent collects game play data of the user if the game state is changed, and reports the game play data for processing.
START

402

DOWNLOAD AND INSTALLED BY USER

404

ACTIVATED BY USER BEFORE STARTING GAME

406

MONITOR GAMING CHARACTERISTICS

408

HAS GAME STATE CHANGED?

410

AGENT COLLECTS USER GAME PLAY DATA

412

STORE DATA IN CLIENT GAME DATA FILE

414

COMMUNICATE WITH SERVER TO STORE AND PROCESS DATA

416

RECEIVE CONFIRMATION OF CREDITS FROM SERVER

418

NOTIFY USER OF ACCOUNT UPDATE (OPTIONAL)

420

END

FIG. 4

START

502

DETECTS COMMUNICATION FROM SOFTWARE AGENT

504

SERVER UPDATE DATA INTO DATABASE

506

CREDIT USER PERSONAL ACCOUNT

508

SEND ACCOUNT UPDATE TO SOFTWARE AGENT/WEB SITE

END

FIG. 5

START

602

USER LOGS IN FOR REDEMPTION

604

RETRIEVE USER ACCOUNT CREDIT STATUS FROM DATABASE

606

DISPLAY ACCOUNT INFORMATION TO USER

608

DETECT SELECTION OF PRODUCTS AND/OR SERVICES BY USER

610

REDEEM PRODUCTS AND/OR SERVICES FOR USER

END

FIG. 6
START

900
TAKE A SNAPSHOT OF OPERATING SYSTEM OF USER'S SYSTEM

902
OBTAIN RULE SET FOR A SET OF CATEGORIES

904
RETRIEVE INDIVIDUAL POINT SCORE FOR NEXT CATEGORY

906
INDIVIDUAL POINT SCORE REACH THRESHOLD?

YES

908
ADD INDIVIDUAL POINT SCORE TO CURRENT CUMULATIVE TOTAL

CURRENT CUMULATIVE TOTAL \(\geq\) TOTAL POINT SCORE?

YES

910
ADDITONAL CATEGORIES?

YES

914
SET GAME STATE TO INACTIVE

NO

918
SET GAME STATE TO IDLE

SET GAME STATE TO ACTIVE

908

NO

916
IS USER'S ACTIVE WINDOW IN FOREGROUND?

YES

920
END
FIG. 10A

GAMER 1010

LOCAL DATA STORES 1008

AGENT 1000

LAUNCH AGENT 1002

START EAMONN ENGINE

(STARTS)

1004

TRACK NEW GAMES

1014

CREATE A NEW GAME OBJECT

1012

NEW GAME STARTED

(RULE SET) (PROVIDES GAME LIST)

1016

(DETECTS) (NEW GAME STARTED)

1018

USERS ACTIONS

_USERS ACTIONS (DETECTS)

1016

(RULE SET) (IS USED TO DETERMINE)

1020

GATHER ENTROPY

GAME STATE

1022

(STATE HAS CHANGED)

1024

ENCRIPT CLIENT GAME DATA

1026

CLIENT GAME DATA

(WRITE DATA TO DISK)

PROCESS DONE

TO FIG. 10B
FROM FIG. 10A

GAMER 1010

LOCAL DATA STORES 1008

AGENT 1000

1028

MANUAL UPDATE CREDITS

(IF ACTIVATED MANUALLY)

1030

TRANSPORT LOCAL DATA

GET NETWORK STATUS

1032

(ONLINE)

CHECK CLIENT GAME DATA THRESHOLD

1034

(CHECK THRESHOLD)

1036

SEND SERVER UPDATES

(EXIT)

THRESHOLD (REACHED)

SEND SERVER UPDATES (END OF UPDATES REACHED)

EXIT TRANSPORT

FIG. 10B
FIG. 13A

START

1300 DETECT USER UPDATE OF RESPECTIVE GAME LIST IN PROFILE

1302 UPDATE USER'S GAME LIST DATA

1304 DETECT SYNCHRONIZATION REQUEST FROM AGENT

1306 SEND GAME LIST AND CORRESPONDING DYNAMIC LIBRARY TO SOFTWARE AGENT

1308 DETECT COMMUNICATION FROM SOFTWARE AGENT

1310 RECEIVE CLIENT GAME DATA WITH SUPPORT FOR GAME IN UPDATED GAME LIST

1312 CREDIT USER PERSONAL ACCOUNT BASED ON PERFORMANCE STATS

1314 SEND ACCOUNT UPDATE TO SOFTWARE AGENT/WEB SITE

END

FIG. 13B

START

1320 DETECT USER STARTING NEW GAME

1322 LOADS CORRESPONDING LIBRARY AND GAME VARIABLE LIST INTO MEMORY

1324 USE GAME VARIABLE LIST TO LOAD ADDITIONAL FUNCTIONS INTO MEMORY

1326 INVOKE FUNCTIONS TO STORE DATA AT SET INTERVAL

1328 SEND CLIENT DATA TO SERVER

END
METHOD AND APPARATUS FOR MONITORING USER GAME-PLAYING BEHAVIOR AND REPORTING GAME PLAY DATA IN A GAMING ENVIRONMENT

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates generally to a gaming environment. In particular, the present invention relates to game play in a gaming environment. Still more particularly, the present invention relates to monitoring user gameplaying behavior and reporting game play data in a gaming environment.

[0003] 2. Description of the Related Art

[0004] With the rise of the Internet, gamers all over the world are able to interactively play games online without restrictions of locations and/or time differences. Via an online gaming community, gamers can share information about games as well as other interests relating to gaming, for example, usability, desired features, and the like. For game companies who manufacture these games, interests of gamers become a useful source of information relating to marketing, development, and production of games. The interests of gamers also generate statistics that help companies to identify sales trends. Examples of useful statistics include sales figures based on geographic, rate, and price to sales ratio.

[0005] Gaming community software, such as GamerMetrics, performs analysis of gamer interests for game companies. GamerMetrics is a product available from IGN Entertainment, Inc. GamerMetrics mainly deals with gamer activities through the phases of the game-buying cycle. GamerMetrics collects factors from a sample of gamers that drive their decisions towards purchasing the game. However, after the game is purchased, no analysis is performed by GamerMetrics because it only monitors gamer activities up to the point of sales. While other online gaming community software provides some data after the point of sales, this data is mostly biased because either a game is being promoted or a fee is charged for playing. In addition, true gaming activities are not verified by current gaming community software. For example, a user with a monthly subscription may be logged into the game server for two days, but only actively played the game for two hours.

[0006] While resale data is important to game companies in determining what type of games to develop, post sales data is equally important in monitoring future sales trends. Post sales data enables analysis of gamers’ post sales behavior and what goes on after gamers purchased the game. Some questions that post sales data may answer include how much the game is being played, how long the game is being played, over what period of time after the point of sale the game is played (days, weeks, months, years), who is playing the game (demographic), and where the game is played (geographic).

[0007] In addition, post sales data helps to provide insight to the popularity and longevity of the game. This post sales information may be useful in determining whether a sequel should be developed, correlations to other compatible games in the market, what type of target audience to focus on, decisions to pursue future development of game titles, detailed demographic targeting for future games, timing for such future developments, as well as budgeting and marketing constraints for developing future games. Currently, no solution exists that provides such post sale data. In addition, no solution exists for monitoring game play data while the gamers are offline. Solutions like GamerMetrics only provide analysis while the games are played online. Furthermore, no solution exists that provides a single-player game play data that is non-biased because other solutions only provide game play data of two or more gamers while they are playing a game that is being promoted.

[0008] Furthermore, the participation of gamers in services like IGN’s GamerMetrics and Gamespy hinder on the fact that the service is provided for a fee. Gamers may participate in these services, regardless of single or multiplayer environments. The point is that the data collected from these gamers’ activities is still being sourced from a network server IGN operates and facilitates. The vast majority of game usage falls into private use, which can be an offline use, online private network use for single and multiplayer gaming.

SUMMARY OF THE INVENTION

[0009] The aspects of the present invention provide a computer implemented method, a data processing system, and a computer program product for monitoring user game play behavior and reporting game play data in a gaming environment. Gaming characteristics of a user are monitored upon a launch of a game. A game state is identified based on the gaming characteristics. The game play data of the user is collected if any change in the game state occurs and the game play data is reported for processing.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

[0011] FIG. 1 depicts a pictorial representation of a network of data processing systems in which the present invention may be implemented;

[0012] FIG. 2 is a block diagram of a data processing system that may be implemented as a server in accordance with an illustrative embodiment of the present invention;

[0013] FIG. 3 is a diagram illustrating interactions between components of the present invention in accordance with an illustrative embodiment of the present invention;

[0014] FIG. 4 is a flowchart of a process for collecting user game play data and crediting users in accordance with an illustrative embodiment of the present invention;

[0015] FIG. 5 is a flowchart of a process for processing user game play data in accordance with an illustrative embodiment of the present invention;

[0016] FIG. 6 is a flowchart of a process for managing spending of account credits by the user in accordance with an illustrative embodiment of the present invention;
FIG. 7 is a use case diagram illustrating use cases performed by a gamer, a software agent, a database server, a Web server, and a chat server in accordance with an illustrative embodiment of the present invention;

FIGS. 8A and 8B are activity diagrams illustrating activities performed by a gamer, a software agent, a server, and a chat server in accordance with an illustrative embodiment of the present invention;

FIG. 9 is a flowchart of a process for determining a game state in accordance with an illustrative embodiment of the present invention;

FIGS. 10A-10C are diagrams illustrating activities and data involved when validating, storing, and transporting user game play data in accordance with an illustrative embodiment of the present invention;

FIGS. 11A-11B are class diagrams illustrating exemplary classes used to implement the aspects of the present invention in accordance with an illustrative embodiment of the present invention;

FIGS. 12A-12B are activity diagrams illustrating chat functionality provided by aspects of the present invention are depicted in accordance with an illustrative embodiment of the present invention;

FIG. 13A is a flowchart of a process for collecting user game play data from a game from the perspective of a server in accordance with an illustrative embodiment of the present invention; and

FIG. 13B is a flowchart of a process for collecting user game play data from a game from the perspective of a software agent in accordance with an illustrative embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIGS. 1-2 are provided as exemplary diagrams of data processing environments in which embodiments of the present invention may be implemented. It should be appreciated that FIGS. 1-2 are only exemplary and are not intended to assert or imply any limitation with regard to the environments in which aspects or embodiments of the present invention may be implemented. Many modifications to the depicted environments may be made without departing from the spirit and scope of the present invention.

With reference now to the figures, FIG. 1 depicts a pictorial representation of a network of data processing systems in which aspects of the present invention may be implemented. Network data processing system 100 is a network of computers in which embodiments of the present invention may be implemented. Network data processing system 100 contains network 102, which is the medium used to provide communication links between various devices and computers connected together within network data processing system 100. Network 102 may include connections, such as wire, wireless communication links, or fiber optic cables.

In the depicted example, server 104 and server 106 connect to network 102 along with storage unit 108. In addition, clients 110, 112, and 114 connect to network 102. Clients 110, 112, and 114 may be, for example, personal computers or network computers. In the depicted example, server 104 provides data, such as boot files, operating system images, and applications to clients 110, 112, and 114. Clients 110, 112, and 114 are clients to server 104 in this example. Server 104 may be, for example, a gamer server that hosts games that are played by users at clients 110, 112, and 114. Network data processing system 100 may include additional servers, clients, and other devices not shown.

In the depicted example, network data processing system 100 is the Internet with network 102 representing a worldwide collection of networks and gateways that use the Transmission Control Protocol/Internet Protocol (TCP/IP) suite of protocols to communicate with one another. At the heart of the Internet is a backbone of high-speed data communication lines between major nodes or host computers, consisting of thousands of commercial, governmental, educational, and other computer systems that route data and messages. Of course, network data processing system 100 also may be implemented as a number of different types of networks, such as for example, an intranet, a local area network (LAN), or a wide area network (WAN). FIG. 1 is intended as an example, and not as an architectural limitation for different embodiments of the present invention.

With reference now to FIG. 2, a block diagram of a data processing system is shown in which aspects of the present invention may be implemented. Data processing system 200 is an example of a computer, such as server 104 or client 110 in FIG. 1, in which computer usable code or instructions implementing the processes for embodiments of the present invention may be located.

In the depicted example, data processing system 200 employs hub architecture including north bridge and memory controller hub (NB/MCH) 202 and south bridge and input/output (I/O) controller hub (SB/ICH) 204. Processing unit 206, main memory 208, and graphics processor 210 are connected to NB/MCH 202. Graphics processor 210 may be connected to NB/MCH 202 through an accelerated graphics port (AGP).

In the depicted example, local area network (LAN) adapter 212 connects to SB/ICH 204. Audio adapter 216, keyboard and mouse adapter 220, modem 222, read only memory (ROM) 224, hard disk drive (HDD) 226, CD-ROM drive 230, universal serial bus (USB) ports and other communications ports 232, and PCI/PCIe devices 234 connect to SB/ICH 204 through bus 238 and bus 240. PCI/PCIe devices may include, for example, Ethernet adapters, add-in cards and PC cards for notebook computers. PCI uses a card bus controller, while PCIe does not. ROM 224 may be, for example, a flash binary input/output system (BIOS).

HDD 226 and CD-ROM drive 230 connect to SB/ICH 204 through bus 240. HDD 226 and CD-ROM drive 230 may use, for example, an integrated drive electronics (IDE) or serial advanced technology attachment (SATA) interface. Super I/O (SIO) device 236 may be connected to SB/ICH 204.

An operating system runs on processing unit 206 and coordinates and provides control of various components within data processing system 200 in FIG. 2. As a client, the operating system may be a commercially available operating system such as Microsoft® Windows® XP (Microsoft and Windows are trademarks of Microsoft Corporation in the
United States, other countries, or both). An object-oriented programming system, such as the Java™ programming system, may run in conjunction with the operating system and provides calls to the operating system from Java™ programs or applications executing on data processing system 200 (Java is a trademark of Sun Microsystems, Inc. in the United States, other countries, or both).

[0034] As a server, data processing system 200 may be, for example, an IBM® eServer™ pSeries® computer system, running the Advanced Interactive Executive (AIX®) operating system or the LINUX operating system (eServer, pSeries and AIX are trademarks of International Business Machines Corporation in the United States, other countries, or both while LINUX is a trademark of Linus Torvalds in the United States, other countries, or both). Data processing system 200 may be a symmetric multiprocessor (SMP) system including a plurality of processors in processing unit 206. Alternatively, a single processor system may be employed.

[0035] Instructions for the operating system, the object-oriented programming system, and applications or programs are located on storage devices, such as HDD 226, and may be loaded into main memory 208 for execution by processing unit 206. The processes for embodiments of the present invention are performed by processing unit 206 using computer usable program code, which may be located in a memory such as, for example, main memory 208, ROM 224, or in one or more peripheral devices 226 and 230.

[0036] Those of ordinary skill in the art will appreciate that the hardware in FIGS. 1-2 may vary depending on the implementation. Other internal hardware or peripheral devices, such as flash memory, equivalent non-volatile memory, or optical disk drives and the like, may be used in addition to or in place of the hardware depicted in FIGS. 1-2. Also, the processes of the present invention may be applied to a multiprocessor data processing system.

[0037] In some illustrative examples, data processing system 200 may be a personal digital assistant (PDA), which is configured with flash memory to provide non-volatile memory for storing operating system files and/or user-generated data.

[0038] A bus system may be comprised of one or more buses, such as bus 238 or bus 240 as shown in FIG. 2. Of course, the bus system may be implemented using any type of communication fabric or architecture that provides for a transfer of data between different components or devices attached to the fabric or architecture. A communication unit may include one or more devices used to transmit and receive data, such as modern 222 or network adapter 212 of FIG. 2. A memory may be, for example, main memory 208, ROM 224, or a cache such as found in NVM/MCH 202 in FIG. 2. The depicted examples in FIGS. 1-2 and above-described examples are meant to imply architectural limitations. For example, data processing system 200 also may be a tablet computer, laptop computer, or telephone device in addition to taking the form of a PDA.

[0039] The aspects of the present invention monitor user game play behavior and report the game play data. In order to generate game play data from users, one aspect of the present invention provides a software agent that users can download and install on their gaming systems. Gaming systems include computer systems, such as data processing system 200 in FIG. 2, or any other game entertainment systems that support downloading or installation of code. Examples of game entertainment systems include PlayStation®2 available from Sony Computer Entertainment America Inc. or XBox® available from Microsoft Corporation. The software agent accurately monitors certain gaming characteristics to determine whether the user is actively playing a game and collects user game play data if the user’s state changes. Examples of gaming characteristics that the software agent monitors include mouse usage, joystick usage, keyboard usage, user CPU utilization, user active window, user memory usage, network connectivity, and the like.

[0040] User game play data is data that reflects how users are playing the game. Before a user starts playing a game, the user activates the software agent to collect game play data. When the software agent determines that the user is actively playing a game based on the gaming characteristics, or that any change in game state has occurred, the software agent collects user game play data and continues to monitor user game play behavior. Alternatively, the software agent may automatically activate when the user starts a game. Examples of user game play data include gameplay duration, time of the day the game is played, the title of the game played, and other performance statistics. Unlike other common gaming community software, the software agent of the present invention may collect both online and offline user game play data.

[0041] User game play data is user game play data that is collected while the software agent is running. The user may be playing a game remotely while connected to the Internet or locally on a personal computer without connecting to the Internet. The software agent of the present invention may collect online user game play data remotely or offline user game play data locally. The user game play data may be used to create dynamic services for purposes of interactive tips and hints as well as a dynamic user interface for purposes of communication amongst users. Dynamic services that are created include, for example, without limitation, competitive leagues, tournament competition, and statistical competitions.

[0042] Any time after the software agent collects online and/or offline user game play data, the software agent may communicate with a server to store and process the user collected data provided that a network connection is available. If network connection is not always available, the game play data is stored and transmitted at a time when the network connection becomes available. In most cases, the network connection is one that is indirectly or directly to the Internet. In these examples, the software agent and the server communicate with each other via a secure communication protocol, for example, simple object access protocol (SOAP) over secure hypertext transfer protocol (HTTPS). SOAP is an established protocol that is based on sending markup language messages between hosts. HTTPS is a secure means for transferring data using the hypertext transfer protocol (HTTP). HTTPS uses a 40-bit encryption or 128-bit encryption. In one illustrative embodiment, aspects of the present invention provide a Web service on top of SOAP that customizes the markup language messages being sent between the server and the client.
The software agent and the server deliver both static and dynamic data to each other. Static data includes agent skin files, a game list, a rule set, and other configuration files for configuring the software agent. Dynamic data includes streaming statistics, a chat buddy list, buddy statuses, local user status (active, idle, or inactive), and real-time performance statistics. An agent skin file is an image file layered over an application window to customize the window’s look and feel. In cases where a network connection is not available, the collected user game play data is accumulated locally in a client game play data file until a network connection is detected. In cases where the network connection is available, the client game play data is still accumulated locally. However, the client game data is actively sent to the server based on defined threshold limits. For example, if the user is online, and has accumulated ten updates, those updates are sent to the server and the client game play data file is reset. This means that the client game data includes game data that is stored to a local device, and/or other game data that is being streamed to the server at real time.

After the server receives the collected user game play data from the software agent, the server processes the data accordingly. The processing of data includes updating the collected game play data into a database. In an illustrative embodiment, the database includes all game play data that is collected for different users. Based on the updated data in the database, another aspect of the present invention manages the user’s personal account. The user’s personal account includes account credits that the user may use to redeem products and/or services in a virtual mall.

The management of the user’s personal account includes crediting the user with account credits and managing the spending of the account credits in a virtual mall. The user may spend their account credits on products and/or services that are offered in the virtual mall. The crediting of account credits may be based on cumulative real-time game play data, for example, a total amount of time spent on the game. The crediting of account credits may also be based on incentives, for example, the number of points scored by the user for the game. In addition to the amount of time spent and the number of points scored, other types of incentives, including kills, levels, mission completion, status, and wins may also be used in crediting the users.

Turning now to FIG. 3, a diagram illustrating interactions between components of the present invention is depicted in accordance with an illustrative embodiment of the present invention. As shown in FIG. 3, software agent 301 of the present invention may be implemented within data processing system 302 or game console 310. Data processing system 302 may be implemented as data processing system 200 in FIG. 2. Game console 310 may be implemented as a game entertainment system, such as PlayStation®2 available from Sony Computer Entertainment America Inc. or XBox® available from Microsoft Corporation.

In order to generate user game play data, software agent 301 monitors certain gaming characteristics to determine whether user 300 is actively playing a game. These gaming characteristics include mouse usage, joystick usage, keyboard usage, user processor utilization, user active window, user memory usage, user network connectivity, and the like. Software agent 301 obtains mouse usage by regularly querying the physical state of the mouse 306. Similarly, keyboard and joystick usage are obtained by regularly querying the physical state of the keyboard 308 and joystick 304. In addition, software agent 301 may obtain usage of other input devices by regularly querying the physical state of the hardware devices, for example, personal digital assistan (PDA) 328 and smart phone 332.

Other gaming characteristics, such as user processor utilization, user active window, user memory usage, and user network connectivity may be determined from an operating system currently executing on data processing system 302 without intruding user 300. The operating system of data processing system 302 provides an interface that allows software agent 301 to extract these gaming characteristics. An example of such an operating system is Windows® XP, a product available from Microsoft Corporation. If software agent 301 is implemented in game console 310, gaming characteristics may include, for example, game controller usage obtained from inputs by user 300 to game console 310 via game controller 312.

Based on the gaming characteristics, software agent 301 determines whether user 300 is actively playing a game. In the illustrative examples, software agent 301 constantly monitors changes in game state, and constantly collects data regarding changes in those states. For example, software agent 301 collects client game data when the state changes from idle to active, from active to inactive, from inactive to idle, and the like. The user game play data is data that reflects how user 300 is playing the game. This user game play data includes the game title being played, a username, the duration of time played, and other statistical and performance data, such as the total number of points scored. Performance statistics may include any relevant measure of success with respect to the game being played.

The game play data collected by software agent 301 is unique in that the user game play data is entirely post sales data that is sampled under unsupervised circumstances. This user game play data is advantageous over data that is currently collected by other game community software, because this data provides information that is not otherwise available after the point of sale. Even if the information is available after the point of sale, it is usually skewed or biased and does not account for active game playing. The user game play data that is collected by software agent 301 not only reflects what currently goes on after the point of sale, but also helps companies in predicting future trends of game playing. The number of units sold can be very different from the number of units played. With software agent 301, the number of units actually played may be determined based on certain gaming characteristics. Thus, useful post sales market research data may be provided to various industry stakeholders, including game media planners or buyers, game publishers, game developers, game industry analysts, and other game industry professionals for analysis.

The post sales data that is provided by aspects of the present invention is unbiased and shows the true nature of post game activities, not just participation through dedicated services provided by vendors. Using an analogy with determining the desires and behavior of consumer eating,
In addition to the uniqueness of the user game play data collected, software agent 301 provides an advantage over current gaming community software. In that software agent 301 may collect user game play data both online and offline. Current gaming community software mostly collects user game play data while the user is playing a game online with another user either over the Internet or across local networks. For user game play data that is collected between users while they are online, this user game data may be skewed or biased and does not reflect active game playing. For user game play data that is collected between users while they are playing across local networks, this user game play data is not readily available for monitoring by outside sources. In addition, current gaming community software fails to collect user game play data for a single-player game that may occur offline, since most gaming community software collects user game play data only during game playing of two or more users.

As the user game play data is collected, software agent 301 determines whether a network connection is available in either data processing system 302 or game console 310. If a network connection is available, software agent 301 sends the collected data accumulated in client game data file 314 on data processing system 302 or game console 310 to server 320 via Internet 316 using a secure communication protocol, such as, SOAP over HTTPS. The software agent 301 sends the collected user game play data accumulated in client game data file 314 to server 320 based on a pre-determined threshold and frequency, for example, every ten minutes. If no network connection is available, the collected user game play data continues to be accumulated in client game data file 314 on data processing system 302 or game console 310 until a network connection is detected. Once a network connection is detected, the data is sent in small batches. Client game data file 314 collects user game play data while user 300 is playing a game offline or online.

Once the collected data is received, server 320 extracts the collected user game play data and processes it. Server 320 processes the collected user game play data by updating the data in database 322 with the existing game play data. Server 320 may send the collected user game play data to database 322 via an Internet connection. Database 322 includes a history of game play data that is collected from different users. Once server 320 updates the collected data into database 322, the data may be used by another aspect of the present invention to perform different business functions.

For example, one business function that may be performed by the aspect of the present invention is to send dynamic statistics derived from the updated data directly to software agent 301 and/or Web pages on data processing unit 302 or game console 310. In this way, user 300 may receive immediate feedback as to the status of the user's personal account.

Another business function that may be performed by the aspect of the present invention is to manage the user's personal account. The personal account includes account credits that the user may use to redeem products and/or services. The aspect of the present invention may credit the user with account credits and manage the spending of the account credits in virtual mall 324 that is provided by another aspect of the present invention.

Virtual mall 324 includes a variety of products and/or services that are available for redemption by the user. Virtual mall 324 may be implemented in a Web server, such as Web server 321. User 300 may interact with virtual mall 324 via data processing system 302. Examples of products and services that are offered in virtual mall 324 include consumer electronics, entertainment software, food or restaurant vouchers, and film related products, such as movie tickets.

As discussed above, the user may spend their account credits on products and/or services that are offered in virtual mall 324. The crediting of account credits may be based on cumulative real time game play data, for example, a total amount of time the user spent playing the game, or incentives, for example, the number of points the user scored in the game. In addition to the above, software agent 301 also facilitates chat functionalities between user 300 and other users by communicating with chat server 326 via Internet 316.

Turning now to FIG. 4, a flowchart of a process for collecting user game play data and crediting users is depicted in accordance with an illustrative embodiment of the present invention. This exemplary process may be implemented in a software agent, such as software agent 301 in FIG. 3. As shown in FIG. 4, from the perspective of the software agent, the process begins when the software agent is downloaded by a user and installed on a data processing system or a game console (step 402).

Next, the software agent is activated by the user before starting a game (step 404). The software agent then monitors certain gaming characteristics (step 406). A determination is then made by the software agent as to whether the game state changes based on the characteristics (step 408). More details regarding how the determination is made by the software agent is discussed in FIGS. 9A and 9B. If the game state changes, the software agent collects that game play data (step 410) and stores the collected data in the client game data file (step 412). More details regarding how the software agent collects user game play data is discussed in FIGS. 13A and 13B. If the game state does not change, the software agent returns to step 406 to continue monitoring the gaming characteristics and step 408 to determine if the game state has changed.

The software agent cumulatively collects user game play data and stores the data in the client game data file. Once the threshold for update count and/or frequency has been met (step 413), the software agent then makes a determination as to whether the user is currently online (step 414). The determination is made based on whether a network connection is available in data processing system 302.
or in game console 310 in FIG. 3. If the user is currently online, the software agent communicates with a server to store and process the collected data (step 410). If the user is offline, and the process returns to step 410 to cumulatively collect user game play data and step 412 to cumulatively store the data.

[0062] The accumulated data may be stored in a client game data file in the user's system, such as data processing system 302 or in game console 310 in FIG. 3. Later, the software agent may request an update of the user's personal account and receive confirmation of credits from the server (step 418). The software agent in turn may notify the user of the account update (step 420). Step 420 is an optional step and the process terminates thereafter.

[0063] Turning now to FIG. 5, a flowchart of a process for processing user game play data is depicted in accordance with an illustrative embodiment of the present invention. This exemplary process may be implemented in a server, such as server 320 in FIG. 3. As shown in FIG. 5, from the perspective of the server, the process begins after the software agent communicates with the server at step 416 in FIG. 4. The process begins with the server detecting a communication from the software agent (step 502). In turn, the server updates the collected data into a database (step 504). The database includes a history of game play data for different users.

[0064] Once the database is updated with the collected data, the server may credit the user's personal account based on the collected data (step 506). The server may credit the user's personal account based on cumulative real time game play data, for example, a total amount of time spent on the game. The crediting of account credits may also be based on incentives, for example, the number of points scored by the user for the game. Later, the user may request an update of the personal account; the server then sends an update of the account to a software agent or a Web site (step 508).

[0065] Once account credits are credited to a user's personal account, the user may either view their credit balance or redeem their account credits for products and/or services in a virtual mall. Another aspect of the present invention manages the spending or redeeming of account credits in that virtual mall. Turning now to FIG. 6, a flowchart of a process for managing spending of account credits by the user is depicted in accordance with an illustrative embodiment of the present invention. This exemplary process may be implemented in a Web server, such as Web server 321 in FIG. 3. As shown in FIG. 6, from the perspective of the Web server, the process begins when the user logs in to the virtual mall for redemption (step 602). Next, the Web server retrieves the data that is updated in the database (step 604). In turn, the Web server displays the account information to the user (step 606). This step may be implemented by displaying advertisement to inform the user that credits are present for redemption. For example, the credits may be given to the user because the amount of time that the user spent playing the game exceeds a certain threshold.

[0066] After the account information is displayed, the Web server may detect the user's selection of products and/or services available in the virtual mall (step 608). In turn, the Web server redeems the selected products and/or services for the user (step 610). Thus, the process terminates thereafter.

[0067] Turning now to FIG. 7, a use case diagram illustrating use cases performed by a gamer, a software agent, a database server, a Web server, and a chat server is depicted in accordance with an illustrative embodiment of the present invention. As shown in FIG. 7, gamer 700, interacts with a game by user input device 702, which include moving a joystick, a keyboard, or a mouse. Gamer 700 may also use other types of input gaming devices, including game controller, without departing the spirit and scope of the present invention.

[0068] Gamer 700 can launch game 701, once agent 708 is activated. Agent 708 monitors the movement of input devices, including joystick, keyboard, and mouse, to determine whether gamer 700 is actively playing the game. These characteristics are known as user inputs and the use case for determining the user inputs by the software agent is referred to as validate entropy 710. In addition to validate entropy 710, agent 708 also tracks other gaming characteristics to determine whether the user is actively playing the game. These characteristics may include active process 712, memory usage 714, and CPU 716 utilization. These characteristics may be obtained from an interface provided by the operating system of data processing system 302 or game console 310 in FIG. 3 that gamer 700 is currently using.

[0069] In determining whether the user is active, idle, or inactive, agent 708 also uses predefined algorithms and artificial intelligence in a rule set file. The rule set file may be implemented as a generic plug-in that agent 708 can easily interface with. A rule set defines a threshold limit for each attribute or characteristic that is monitored by the software agent. By defining a threshold limit for each attribute, game companies or resources may specify different criteria for monitoring user behavior.

[0070] The rule set may be defined as GameName, GameID/#, threshold_0, threshold_1, ..., threshold_N. GameName indicates the name of the game being monitored. GameID/# indicates a random game code or "tag" of the game being monitored. Threshold_0 indicates a point weight for attribute_0 while threshold_1 ... threshold_N indicate a point weight from attribute_1 to attribute_N.

[0071] An example of a rule set may be "h12.exe", 4, 40, 2, 6, 7, 10 ... This means that for a game named "h12.exe" with a GameID/#4, the threshold limit for attribute_0 is 40, the threshold limit for attributed is 2 and so on. Each attribute is mapped to a function that has knowledge of how to determine if the attribute is good or bad for a particular cycle. For example, in Windows® environment, a Windows-Zorder attribute is used to indicate a program is in the Windows® foreground. The Windows-Zorder attribute is mapped to a function named "calWindowZorderWin32" which returns true if the attribute’s threshold point score should be added to the total point score, or false if the attribute’s threshold point score should not be added. The total point score is a predefined or static point score that any game must reach in order to consider the user as “active”. If the "calWindowZorderWin32" function returns false, the user is considered “inactive”. However, if no user input is detected in validate entropy 710, the user is considered “idle”. In an illustrative embodiment, the “inactive” status takes precedent over the “idle” status.

[0072] Based on the rule set and validate entropy 710, agent 708 determines whether the user is active, idle, or
inactive. Agent 708 then sets a game state 718 and updates client game data 720. The above use cases repeat until the collected game data reaches a predetermined threshold limit, whereas the agent either sends the updates if the client is online, or continues to accumulate client game data if the user is offline. Once updates are sent, agent 708 resets the local client time data 732 and the Web server resets the client time data on Web server 734. If the agent requests specific dynamic statistical data 730, the database server 728 will send it to the local client. Furthermore, gamer 700 may request update user credits 736. In this case, provided by the aspects of the present invention, the agent will send current accumulated local client game data to Web server 734 and request a current account balance be displayed to gamer 700.

[0073] With the aspects of the present invention, gamer 700 may change agent configuration 722, for example, change functionality or options available to gamer 700. Gamer 700 may also define which games the agent is capable of monitoring by updating game list 724. When agent configuration 722 or the game list 724 is modified, the agent functionality is also updated (update agent process 726), which may affect the data sent to the database server 728. Furthermore, agent 708 manages chat functions 738 when gamer 700 requests chat 740. The chat functions include process chat message 742 with chat server 744, such as chat server 326 in FIG. 3, and process external chat 746 by communicating with external chat servers via chat gateway 748.

[0074] Turning now to FIGS. 8A-8B, activity diagrams illustrating activities performed by a gamer, a software agent, a server, and a chat server are depicted in accordance with an illustrative embodiment of the present invention. The activities in FIGS. 8A and 8B reflect activities that are involved when performing use cases as described in FIG. 7 above. As shown in FIG. 8A, gamer 800 first launches software agent 801, client/agent 802 verifies the connectivity 806 to determine whether agent 800 is connected to server 808. If gamer 800 is connected to server 808, gamer 800 logs in 812 and server 808 receives a prompt and is displayed 814 asking a question 800 whether to continue logging in. After gamer 800 logs in, client/agent 802 verifies the credentials 816 of gamer 800, if the user is online, or remotely if they are online. After gamer 800 logs in, gamer 800 may select start game 818. Responsive to receiving the selection, client/agent 802 executes launch Game.exe 820 to initiate the game and monitor gaming characteristics. Activities 818 and 820 are involved when performing use case launch game 701 in FIG. 7. As gamer 800 performs gaming activities 822, client/agent 802 tracks various processes 824, including CPU utilization, memory usage and the like, and performs validate entropy 826 from user inputs. More details regarding activities 824 and 826 are discussed in FIGS. 9A and 9B. A determination is then made by client/agent 802 as to whether gamer 800 is active, inactive, or idle based on the tracked processes and validated entropy. In turn, client/agent 802 sets game state 828 reflecting whether gamer 800 is active, inactive, or idle. Once the game state is set, client/agent 802 updates client game data 830. Activities 824, 826, 828, and 830 are involved when performing use cases 710, 712-716, 718, and 720 in FIG. 7 respectively.

[0075] When online, gamer 800 may manually trigger client/agent 802 to send a request credit update 832 to their personal account. Upon receiving the request, client/agent 802 sends credit update 834 to server 808, which processes credit update 838 by updating a database. After the credit update is processed, server 808 sends a successful credit update message to client/agent 802, who can display the receive credit update 836 locally. Activities 832, 834, 836, and 838 are involved when performing update user credits use case 736 in FIG. 7.

[0077] Turning now to FIG. 8B, if gamer 800 is connected to server 808, gamer 800 may request an update of game list 840. The user performs the update of game list through a Web browser in the user’s online profile. Once the game list is updated, server 808 updates supported game list 846 that is stored in server 808 and sends the updated game list file to client/agent 802, which stores the updated game list file locally 844. Activities 840, 844, and 846, are involved when performing update game list 724 use case and update agent process 726 in FIG. 7.

[0078] After gamer 800 requests to synchronize agent 848 with server 808, server 808 receives synchronize request 850 and sends static data 852 to client/agent 802, who receives static data 854. Static data may include a rule set, a game list, and other configurations. In addition to requesting static data, client/agent 802 may also request dynamic data 856 from server 808. Dynamic data includes streaming statistics, chat buddy list, buddy status, local user status, and real time performance statistics. Upon receiving the request for dynamic data, server 808 sends dynamic data 858 to client/agent 802, which receives dynamic data 860. Activities 860-860 are involved when performing send dynamic data use case 730 in FIG. 7.

[0079] In addition to the above, client/agent 802 also facilitates chat activities. If gamer 800 logs in for chat (chat login) 862, client/agent 802 requests a new password 864 from server 808. Server 808 assigns new password request 866 to client/agent 802. After client/agent 802 receives the new password from server 808, the new password is sent to client/agent 802 as the password is typed in by client/agent 802. This password is referred to as a current chat password of client/agent 802. Chat server 810 in turn validates the new password 870 and notifies client/agent 802 that a chat is ready 872. Subsequently, client/agent 802 manages various chat functions 874 while chat server 810 processes chat 876. Activities 862-876 are involved when performing use cases 738-742 in FIG. 7.

[0080] At any time, if gamer 800 wants to change configuration 878 of client/agent 802, gamer 800 may modify some configuration options locally, for example, an agent skin file. Local config options 880 can be changed regardless of network connection status and do not require access to the user’s online profile. Most configuration options of client/agent 802 are accessed through the gamers online profile. When a change occurs, server 808 then processes configuration update 882 and stores configuration changes 884. Activities 878-884 are involved when performing change agent configuration 722 use case and update agent process 726 use case in FIG. 7.

[0081] Turning now to FIG. 9, a flowchart of a process for determining a game state is depicted in accordance with an illustrative embodiment of the present invention. This exemplary process may be implemented in a software agent, such as software agent 301 in FIG. 3. As shown in FIG. 9, from
the perspective of the software agent, the process begins when the software agent takes a snapshot of the operating system of the users' system (step 900). The software agent may perform this step periodically during user game play, for example, every minute, and the snapshot may be saved in user's system memory.

[0082] Next, the software agent obtains a rule set for a set of categories (step 902). These categories include processor usage, memory usage, network connectivity, window Z order, and other user inputs. A rule set defines a threshold limit for each category that is being monitored by the functions. From the snapshot of the operating system, the software agent then retrieves an individual point score that is assigned to the next category in the set of categories (step 904). For each category, an individual point score is assigned, for example, processor usage may be assigned a point score of 40 and window Z order may be assigned a point score of 35.

[0083] The software agent then makes a determination as to whether the individual point score reaches the threshold limit as defined for each category (step 906). If the individual point score reaches the threshold limit, the individual point score is added to the current cumulative total (step 908). However, if the individual point score does not reach the threshold limit, the process continues to step 914.

[0084] After step 908, software agent then makes a determination as to whether the current cumulative total is greater than or equal to the predefined total point score for an active state (step 910). The total point score is a predefined or static point score that any game must reach in order to consider the user as “active”. If the current cumulative total is greater than or equal to the predefined total point score for an active state, the software agent sets the game state to “active” (step 912) and the process terminates thereafter. “Active” state means that the user is actively engaged in game playing. If the current cumulative total is not greater than or equal to the predefined total point score for an active state, the process continues to step 914.

[0085] At step 914, a determination is made by the software agent as to whether additional categories are present in the set of categories. If so, the process returns to step 904 to retrieve the next category from the set of categories. Otherwise, a determination is made by the software agent as to whether the user’s active window is in the foreground (step 916). This step is performed by examining the snapshot of the operating system. If the active window is in the foreground, the software agent sets the game state to “idle” (step 918). An example of an “idle” status is when a user completes a level, there may be a clip shown for a number of minutes. Thus, while the processor and memory may be active during the number of minutes, there may be no movement made by the user. If the active window is not in the foreground, the software agent sets the game state to “inactive” (step 920). “Inactive” state means that the user is not currently engaged in game playing. Thus, the process terminates thereafter.

[0086] Turning now to FIGS. 10A-10C, diagrams illustrating activities and data involved when validating, storing, and transporting user game play data are depicted in accordance with an illustrative embodiment of the present invention. FIG. 10A illustrates data flow and validation of user game play data on the local client.
is updated, a game update confirmation 1052 is returned to agent 1000. Agent 1000 continues to check update data 1053 and send game updates 1054 to Web server 1040 until all updated client game data 1048 in the client game data file is sent. Once all the game data is sent, agent 1000 closes server session 1056 by sending a goodbye message to Web server 1040. Web server 1040 in turn closes session 1058, which includes closing the connection with database server 1051, sends a goodbye message to agent 1000, and exits the process.

[0092] As discussed above, the data in database server 1051 may be used for various business functions, which include sending dynamic statistics 1062 to agent 1000, managing gamers’ bank account credits 1064, and managing spending of account credits in a virtual mail using shopping cart 1060.

[0093] Turning now to FIGS. 11A and 11B, class diagrams illustrating exemplary classes used to implement the aspects of the present invention are depicted in accordance with an illustrative embodiment of the present invention. As shown in FIG. 11A, a game may launch a game via user interface 1124 by invoking launchGame 1118 function, invoke a chat using invokeChat 1114 function of chat 1108, and view a list of buddies by invoking viewBuddies 1110 function of chat 1108. In addition, a game may update any options available to them in user interface 1124. When the game synchronizes with the software agent, updates to static content 1170 or dynamic content 1172 will be requested by the software agent and then sent by the Web server, such as Web server 1040 in FIG. 10C.

[0094] Eamonn user class 1100 holds user information, such as userName 1102 and password 1104. When the software agent is launched, Eamonn engine 1101 is started to track new games, process entropy, determine game state, transport local data, and the like. Eamonn engine 1101 tracks new games based on a predefined list of supported games. This list contains the game name, id, and ruleset 1134. Upon detecting that a new game is started by a game, Eamonn engine 1101 creates a game object 1106 as shown in FIG. 11A if the game started is in the list. In FIG. 11A, game object 1106 includes a game state 1130, which indicates whether the game is active, idle, or inactive. When the software agent detects users actions initiated by the game, process entropy 1132 obtains gaming characteristic data, such as mouse movement, keyboard movement, joystick movement, CPU utilization, memory usage, network connectivity and the like. Process entropy 1132 monitors mouseState 1136, keyboardState 1138, joystickState 1140, as well as cpuUtilization 1142, memoryUsage 1144, pid 1146, windowLevel 1148, networkConnectivity 1150, and the like. In addition to process entropy 1132, ruleset 1134 is also used by Eamonn engine 1101 to determine a game state. Ruleset 1134 defines threshold for each attribute that is being monitored, for example, windowOrder 1137, userInputs 1139, cpuUsage 1141, memoryUsage 1143, and networkConnectivity 1145.

[0095] If Eamonn engine 1101 determines that there has been a state change, it collects client game data 1152, which includes userName 1154, computerGuid 1158, startTime 1160, endTime 1162, and gameState 1164. However, Eamonn engine 1101 may collect other client game data, such as kills, points, level, mission completion, status, and wins. Once client game data 1152 is accumulated, Eamonn engine 1101 sends the data to a server via a secure communication protocol, for example, GSOAP 1174 in FIG. 11B, if a network connection is available at a predefined frequency or if a threshold limit is met. The server in turn updates the data into database 1176 in FIG. 11B via an Internet connection.

[0096] Turning now to FIGS. 12A and 12B, activity diagrams illustrating chat functionality provided by aspects of the present invention are depicted in accordance with an illustrative embodiment of the present invention. As shown in FIG. 12A, agent checks user preferences to determine whether to sign into a chat at startup. If user preferences indicate to sign into a chat at startup, agent requests chat password 1200 from Web server. Otherwise, the chat client remains idle 1202 and the user manually logs into the Web server and requests chat password 1200.

[0097] The Web server sends chat password 1204 to the agent and the client accepts password 1206. After the agent receives the new password from the Web server, the agent sends the new password to the chat server for authentication 1208 as if the password is typed in by the client. This password is referred to as a current chat password of the client. Chat server authenticates the user by validating new password 1210. If authentication is successful, a handshake is complete 1212 between the agent and the chat server. The agent may log off 1214 the chat server and exits the chat or selecting a chat function. However, if authentication is not successful, error message 1216 is returned to the user and a retry may occur.

[0098] Aspects of the present invention provide a plurality of chat functions. As shown in FIG. 12B, these chat functions include joining group chat 1218, starting group chat 1220, blocking another user 1222, removing a buddy from subscription 1224, accepting a new subscription request 1226 from another user, adding another user to subscription 1228, sending a message to another user 1230, and receiving messages 1232 from another user.

[0099] If the user receives a message 1232 from another user, the agent determines whether the other user is blocked, on the roster, or not on a roster. If the user is blocked, the message is dropped 1234. If the user is on the roster, the message is displayed 1238. If the user is not on the roster, a prompt is displayed 1236 to ask whether the user wants to accept or reject message. If the user accepts the message, the message is displayed 1238. Otherwise, the message is dropped 1234.

[0100] Turning now to FIG. 13A, a flowchart of a process for collecting user game play data from a game from the perspective of a server is depicted in accordance with an illustrative embodiment of the present invention. This exemplary process may be implemented in a server, such as server 320 in FIG. 3. As shown in FIG. 13A, the process begins, from the perspective of the server, when the server detects a user update of respective game list in the user’s online profile (step 1300). The server then updates the user’s game list data on the server (step 1302). A request of synchronization is received from the agent (step 1304). The agent then sends the synchronization request to obtain the new game list locally. The server in turn sends the game list along with dynamic libraries that are associated with the updated games in the game list to the agent (step 1306). Each game that is
supported may potentially have a different dynamic library. If the agent is never synchronized, a new game list and new dynamic libraries will never be sent. Later, the server detects a communication request from the software agent (step 1308). The server receives client game data updates from the agent with support for specific games on the updated game list (step 1310).

[0101] The server can then credit the personal account of the user with credits based on performance incentives (step 1312), such as the total points scored, in addition to credits that are given based on cumulative real-time game play data, such as an amount of time the user spent playing the game. The server can then send an account update to the software agent either by user request or for dynamic statistics (step 1314), such that the user may view the status of the account. Thus, the process terminates thereafter.

[0102] Turning now to FIG. 13B, a flowchart of a process for collecting user game play data from a game from the perspective of a software agent is depicted in accordance with an illustrative embodiment of the present invention. This exemplary process may be implemented in a software agent, such as software agent 301 in FIG. 3. As shown in FIG. 13B, the process begins when the software agent detects that the user is starting a new game (step 1320). Next, the software agent loads a corresponding dynamic library and a game variable list for the game into memory (step 1322). The game variable list includes variables that the game constantly monitors. This list may be supplied by the game developers.

[0103] After loading the variables into memory, the software agent uses the game variable list to load additional functions into memory (step 1324). Each game variable has a corresponding function in the dynamic library that can calculate the variable. An example of a game variable and corresponding function is WindowZorder and "callWindowZorderWin32" as discussed above.

[0104] At a set interval, the software agent calls the corresponding function to store the data that is collected by the function (step 1326). The data may be encrypted and stored in a client game data file or in a separate file. If a network connection is available, and the threshold is reached, the software agent sends the client game data to the server (step 1328), which includes data that is collected by the function. Thus, the process terminates thereafter.

[0105] In summary, aspects of the present invention provide a model that collects online and offline user game play data and credits the users according to the collected data. The user game play data is post sales market research data that is otherwise skewed or biased. In addition to collecting game play data, the present invention provides a software agent that perform various functionalities, including user account management, chat management, game list management, and the like. Furthermore, aspects of the present invention provide game play data for a single-player game play that is non-biased. Furthermore, another aspect of the present invention involves how the company utilizes the key aspects of the data to provide unique services, for example, dynamic data to initiate an interactive environment for the users, competitive analysis amongst users, as well as data used to promote dynamically efficient targeted advertisements within live games.

[0106] The invention can take the form of an entirely hardware embodiment, an entirely software embodiment, or an embodiment containing both hardware and software elements. In a preferred embodiment, the invention is implemented in software, which includes but is not limited to firmware, resident software, microcode, etc.

[0107] Furthermore, the invention can take the form of a computer program product accessible from a computer usable or tangible computer-readable medium providing program code for use by or in connection with a computer or any instruction execution system. For the purposes of this description, a computer usable or computer readable medium can be any apparatus that can contain, store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device.

[0108] The medium can be an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system (or apparatus or device) or a propagation medium. Examples of a computer-readable medium include a semiconductor or solid-state memory, magnetic tape, a removable computer diskette, a random access memory (RAM), a read-only memory (ROM), a rigid magnetic disk and an optical disk. Current examples of optical disks include compact disk—read only memory (CD-ROM), compact disk—read/write (CD-R/W), and digital video disc (DVD). The medium may be tangible or intangible depending on the implementation.

[0109] A data processing system suitable for storing and/or executing program code will include at least one processor coupled directly or indirectly to memory elements through a system bus. The memory elements can include local memory employed during actual execution of the program code, bulk storage, and cache memories which provide temporary storage of at least some program code in order to reduce the number of times code must be retrieved from bulk storage during execution.

[0110] Input/output or I/O devices (including but not limited to keyboards, displays, pointing devices, etc.) can be coupled to the system either directly or through intervening I/O controllers.

[0111] Network adapters may also be coupled to the system to enable the data processing system to become coupled to other data processing systems or remote printers or storage devices through intervening private or public networks. Modems, cable modems, and Ethernet cards are just a few of the currently available types of network adapters.

[0112] The description of the present invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or limited to the invention in the form disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art. The embodiment was chosen and described in order to best explain the principles of the invention, the practical application, and to enable others of ordinary skill in the art to understand the invention for various embodiments with various modifications as are suited to the particular use contemplated.

What is claimed is:

1. A computer implemented method for monitoring user game play behavior and reporting user game play data in a gaming environment, the computer implemented method comprising:
responsive to a launch of a game by a user, monitoring gaming characteristics of the user;

identifying a game state based on the gaming characteristics;

collecting game play data of the user if the game state is changed; and

reporting the game play data for processing.

2. The computer implemented method of claim 1, wherein the gaming characteristics include user mouse movement, user joystick movement, user keyboard movement, user active process, memory usage, and processor utilization.

3. The computer implemented method of claim 1, wherein the monitoring, identifying, collecting, and reporting steps are performed by a software agent.

4. The computer implemented method of claim 1, wherein the identifying step comprises:

validating gaming characteristics based on user inputs; and

validating gaming characteristics based on a rule set.

5. The computer implemented method of claim 1, wherein the rule set comprises a threshold limit for each characteristic that is monitored.

6. The computer implemented method of claim 1, wherein the reporting step comprises:

accumulating the game play data in a client game data file on the client; and

sending the game play data to a server via a secure communication protocol if a network connection is available on the client and a threshold is reached.

7. The computer implemented method of claim 6, further comprising:

updating the game play data into a database; and

receiving dynamic data from the database, wherein the dynamic data comprises streaming statistics, a chat subscription, chat buddy statuses, a local user status, and real time performance statistics.

8. The computer implemented method of claim 1, wherein the game play data comprises an amount of time the user spent playing the game, a time of day, a game title, and performance statistics.

9. The computer implemented method of claim 1, further comprising:

reading a list of supported games from a game list file, wherein the game list file is created and modified as the user updates the list of supported games in an online profile; and

creating a new game object from the game list for the game to launch the game.

10. The computer implemented method of claim 1, further comprising:

managing chat functions initiated by the user, wherein the chat functions include joining a group chat, starting a group chat, blocking another user, removing another user from a subscription, adding another user to a subscription, sending a message to another user, and receiving a message from another user.

11. A gaming system for monitoring user game play behavior and reporting user game play data, the gaming system comprising:

a bus,

a storage device, wherein the storage device contains computer usable code;

a communications unit connected to the bus;

a user input device; and

a processing unit connected to the bus, wherein the processing unit executes the computer usable code to monitor gaming characteristics of the user responsive to a launch of a game by a user; identify a game state based on the gaming characteristics; collect game play data of the user if the game state is changed; and report the game play data for processing.

12. The gaming system of claim 11, wherein the processing unit, in executing the computer usable code to identify a game state based on the gaming characteristics, executes the computer usable code to validate gaming characteristics based on user inputs; and validate gaming characteristics based on a rule set.

13. The gaming system of claim 11, wherein the processing unit, in executing the computer usable code to report the game play data for processing, executes the computer usable code to accumulate the game play data in a client game data file on the client; and send the game play data to a server via a secure communication protocol if a network connection is available on the client and a threshold is reached.

14. The gaming system of claim 14, wherein the processing unit, in executing the computer usable code to report the game play data for processing, further executes the computer usable code to update the game play data into a database; and receive dynamic data from the database, wherein the dynamic data comprises streaming statistics, a chat subscription, chat buddy statuses, a local user status, and real time performance statistics.

15. The gaming system of claim 11, wherein the processing unit further executes the computer usable code to read a list of supported games from a game list file, wherein the game list file is created and modified as the user updates the list of supported games in an online profile; and create a new game object from the game list for the game to launch the game.

16. The gaming system of claim 11, wherein the processing unit further executes the computer usable code to manage chat functions initiated by the user, wherein the chat functions include joining a group chat, starting a group chat, blocking another user, removing another user from a subscription, adding another user to a subscription, sending a message to another user, and receiving a message from another user.

17. A computer program product comprising:

a computer usable medium having computer usable program code for monitoring user game play behavior and reporting user game play data in a gaming environment, said computer program product comprising:

computer usable program code for monitoring gaming characteristics of the user responsive to a launch of a game by a user;
computer usable program code for identifying a game state based on the gaming characteristics;

computer usable program code for collecting game play data of the user if the game state is changed; and

computer usable program code for reporting the game play data for processing.

18. The computer program product of claim 17, wherein the computer usable program code for identifying a game state based on the gaming characteristics comprises:

computer usable program code for validating gaming characteristics based on user inputs; and

computer usable program code for validating gaming characteristics based on a rule set.

19. The computer program product of claim 17, further comprising:

computer usable program code for updating the game play data into a database; and

computer usable program code for receiving dynamic data from the database, wherein the dynamic data comprises streaming statistics, a chat subscription, chat buddy statuses, a local user status, and real time performance statistics.

20. The computer program product of claim 17, further comprising:

computer usable program code for managing chat functions initiated by the user, wherein the chat functions include joining a group chat, starting a group chat, blocking another user, removing another user from a subscription, adding another user to a subscription, sending a message to another user, and receiving a message from another user.