TOURNAMENT MATCHUPS FOR A MULTIPLAYER ENVIRONMENT

Inventors: James Coliz, Krikland, WA (US); Jeffrey B. Spradling, Issaquah, WA (US); Brian Spanton, Bellevue, WA (US); Scott Plette, Redmond, WA (US); Mark Edmonds, Coleorton (GB)

Correspondence Address:
BANNER & WITCOFF, LTD.
ATTORNEYS FOR CLIENT NOS. 003797 & 013797
1100 13th STREET, N.W.
SUITE 1200
WASHINGTON, DC 20005-4051 (US)

Assignee: Microsoft Corporation, Redmond, WA

Filed: Feb. 16, 2006

Publication Classification

Int. Cl.
A63F 9/24 (2006.01)

U.S. Cl. 463/42

ABSTRACT

Multiplayer tournaments may be established, and then automatically executed by tournament server devices to help provide users with quality matchups against players of similar skill. Tournaments may be defined by an administrator, and then automatically instantiated any number of times to accommodate demand by players. Some tournaments may dynamically adjust tournament parameters to accommodate a variable number of qualifiers entering the tournament. Tournaments may perform ranking, before and/or during the tournament, based on a calculation of a ratio of player points to a total number of points the player could have received. User interface displays may also be provided to allow players to view details of tournaments for which they have qualified, details on their tournament competitors, or search for tournaments satisfying the user’s desired criteria.
Fig. 2
Fig. 4
START

601 ADMINISTRATOR LOGIN

602 DEFINE TOURNAMENT PARAMETERS

603 TRIGGER?

604 INSTANTIATE TOURNAMENT

605 CONDUCT TOURNAMENT

606 CLIENTS REPORT TOURNAMENT RESULTS TO TOURNAMENT SERVER AND LEADERBOARD

607 RESULT ANOMALY?

608 QUERY ARBITRATED LEADERBOARD

609 CORRECT LEADERBOARD

610 CONCLUDE TOURNAMENT?

611 CONCLUDE TOURNAMENT
<table>
<thead>
<tr>
<th>LIST 3</th>
<th>ROUND 1 SETTINGS</th>
<th>ROUND 2 SETTINGS</th>
<th>ROUND 3 SETTINGS</th>
<th>ROUND 4 SETTINGS</th>
<th>ROUND 5 SETTINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST 2</td>
<td>ROUND 1 SETTINGS</td>
<td>ROUND 2 SETTINGS</td>
<td>ROUND 3 SETTINGS</td>
<td>ROUND 4 SETTINGS</td>
<td>ROUND 5 SETTINGS</td>
</tr>
<tr>
<td>LIST 1</td>
<td>ROUND 1 SETTINGS</td>
<td>ROUND 2 SETTINGS</td>
<td>ROUND 3 SETTINGS</td>
<td>ROUND 4 SETTINGS</td>
<td>ROUND 5 SETTINGS</td>
</tr>
</tbody>
</table>

**Figure 7**

701
703
704
702
START

PLAYER REGISTRATION

INITIAL PLACEMENT OF ORDER

DETERMINE NUMBER OF PLAYERS PER GROUP

NEW GROUP

ADD HIGHEST UNASSIGNED TO GROUP

CHECK NEXT UNASSIGNED PLAYER

NEXT PLAYER RECENT?

ADD TO GROUP

GROUP FULL?

REMAINING UNASSIGNED?

Figure 8a
Figure 8b
### ROUND 1 RANKINGS

<table>
<thead>
<tr>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAYER A</td>
</tr>
<tr>
<td>PLAYER B</td>
</tr>
<tr>
<td>PLAYER C</td>
</tr>
<tr>
<td>PLAYER D</td>
</tr>
<tr>
<td>PLAYER E</td>
</tr>
<tr>
<td>PLAYER F</td>
</tr>
<tr>
<td>PLAYER G</td>
</tr>
<tr>
<td>PLAYER H</td>
</tr>
<tr>
<td>PLAYER I</td>
</tr>
<tr>
<td>PLAYER J</td>
</tr>
<tr>
<td>PLAYER K</td>
</tr>
<tr>
<td>PLAYER L</td>
</tr>
</tbody>
</table>

### ROUND 2 RANKINGS

<table>
<thead>
<tr>
<th>Player</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLAYER A</td>
</tr>
<tr>
<td>PLAYER B</td>
</tr>
<tr>
<td>PLAYER C</td>
</tr>
<tr>
<td>PLAYER D</td>
</tr>
<tr>
<td>PLAYER E</td>
</tr>
<tr>
<td>PLAYER F</td>
</tr>
<tr>
<td>PLAYER G</td>
</tr>
<tr>
<td>PLAYER H</td>
</tr>
<tr>
<td>PLAYER I</td>
</tr>
<tr>
<td>PLAYER J</td>
</tr>
<tr>
<td>PLAYER K</td>
</tr>
<tr>
<td>PLAYER L</td>
</tr>
</tbody>
</table>

---

**Figure 8c**
START

901 DEFINE TOURNAMENT

902 ANNOUNCE QUALIFICATION

903 QUALIFICATION PERIOD

904 DIVIDE QUALIFIERS INTO LEVELS

905 SCHEDULE AND NOTIFY QUALIFIERS

906 SEED N TOURNAMENT LEVELS

907 PLAY ROUNDS OF TOURNAMENT

END

Figure 9
START

1001
SET CURRENT ROUND TO FINAL ROUND

1002
FILL CURRENT ROUND

1003
DETERMINE NUMBER OF FEEDER MATCHES NEEDED TO FILL CURRENT ROUND

1004
DISTRIBUTE CURRENT ROUND PLAYERS AMONG ROUNDS IN FEEDER MATCHES OF EARLIER ROUND

1005
FILL REMAINDER OF FEEDER MATCHES IN EARLIER ROUND

1006
ALL ENTRANTS PLACED?

1007
CURRENT ROUND = ONE ROUND EARLIER

END

Figure 10b
TOURNAMENT MATCHUPS FOR A MULTIPLAYER ENVIRONMENT

BACKGROUND

[0001] Online computer gaming has quickly grown into a key component of today’s video game market, as there are now more and more opportunities to play against others. For example, on the XBOX LIVE® (Microsoft Corporation, Redmond, Wash.) online service, it is possible for video game players to play against other players from all over the world. With such a large pool of potential players, there is a correspondingly large range of player abilities, ranging from the first-time or occasional player to the daily, devoted fan, and even on to the ranks of professional video game players.

[0002] This wide range of player abilities has introduced a new problem. Many online games are competitive, in that players compete against one another to achieve game objectives and gain a level of ascendancy over their competitors. Such competitive games are often only enjoyable when the various players are close to one another in skills and abilities. If there is too great a discrepancy in skill level, the dominant-skilled player will not be challenged, while the weaker-skilled player will experience frustration at constantly being beaten. Typical online games, however, collect players as they join game sessions, with no regard for the players’ abilities or how close the players are to one another in terms of skill.

[0003] Some organized competitions, such as tournaments, may attempt to match players of equal skill in a final round. For example, the NCAA® (National College Athletics Association) basketball tournament takes sixty-five teams, and seeds the teams such that (assuming no upsets) its two semifinal games pit the top four teams against one another (i.e., the four number 1 seeds). However, the path of each number 1 seed through the earlier rounds of the tournament is riddled with imbalanced matchups. For example, the first round of the NCAA® basketball tournament pits the top seed in a bracket against the lowest seed in a bracket (16th seed). Additionally, the NCAA® arrangement requires a fixed, limited lineup of teams (the 65 teams in seeded order), and would not easily translate into a video game community having thousands (or millions) of players, and players who may drop out or come in at any time.

SUMMARY

[0004] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

[0005] As will be described in greater detail below, the present application includes a customizable competition structure. Tournament parameters may be defined in advance, and may be automatically instantiated by the game’s communication network so that tournaments can be created as needed and exist indefinitely, without the need for constant human supervision. This definition may be done using, for example, an Internet page interface, or by an end user’s video game user interface.

[0006] A tournament administrator may log in with a tournament server to define the tournament, entering tournament parameters. The tournament may be instantiated one or more times, and the separate instances may individually, and automatically, communicate with the various participants, manage game session results, and determine tournament winners.

[0007] In some aspects, scoring arbitration may be used to resolve anomalies in scores reported by the various game consoles in the tournament. For example, a tournament server may compare its received results with results received by a separate leaderboard process, where the separate leaderboard process may use more secured transmissions with the game consoles to help verify communications.

[0008] In some aspects, tournaments may run according to a schedule, and may be configured to run one after another (e.g., substantially continuously).

[0009] In some aspects, a window factor can identify an amount of time that must elapse before players can be reunited in a game session, to minimize effect of having same players play each other throughout a tournament. Also, a new player factor can be used to adjust scores of newly-entered participants.

[0010] In some aspects, tournament parameters and game settings may be downloaded to a game console to automatically configure the player’s machine for the next round of a tournament.

[0011] In some aspects, a qualification period may precede the tournament, during which players may attempt to qualify by posting their corresponding scores. Players may be notified of their qualification status, and may be notified again if the status changes due to subsequent players’ qualification attempts, and may be given an option to make another attempt at qualification. A tournament may be instantiated multiple times to accommodate differing levels of qualifiers.

[0012] In some aspects, tournaments may be dynamically adjusted to accommodate variable numbers of qualifying tournament entries. The tournament parameters, such as number of winners per match, number of players per match, number of rounds, number of players in the tournament’s final round, and others, may be varied so that a desired number of qualifiers may participate in the tournament. Entrants may be ranked using a ratio of their total score to the total possible score for rounds that they have played, and that ranking may be used in later groupings of players.

[0013] Tournament user interfaces may be displayed to allow user navigation through the tournament process. The user may play an interactive, online multiplayer and multi-round video game, and view a list of tournaments for which he/she has qualified. Other options may be offered as well, such as searching for tournaments that satisfy user-desired criteria, or displaying tournament and/or competitor details.

[0014] These and other aspects will be described in greater detail below.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] FIG. 1 illustrates a game console that may be used to implement various features described herein.
FIG. 2 illustrates components that may be used in the console shown in FIG. 1.

FIG. 3 illustrates how various consoles, and other elements, may be interconnected to implement features described herein.

FIG. 4 illustrates an example network configuration using game consoles.

FIG. 5 illustrates an example computing device that may be used to implement features described herein.

FIGS. 5a-n illustrate example software elements and concepts for implementing features described herein.

FIG. 6 illustrates an example process for conducting tournaments.

FIG. 7 illustrates an example of a tournament data structure.

FIGS. 8a-b illustrate an example process for a type of tournament, and FIG. 8c illustrates an example progression of ranking and reranking.

FIG. 9 illustrates another example process for an alternative type of tournament.

FIGS. 10a and 10b illustrate an example seeding bracket and process for seeding.

FIG. 11 illustrates another example seeding bracket.

FIG. 12 illustrates an example relationship chart showing how various user interface screens may be used on a player’s console.

DETAILED DESCRIPTION

In the following description of the various aspects, reference is made to the accompanying drawings, which form a part hereof, and in which is shown by way of illustration various features described herein may be practiced. It is to be understood that other embodiments may be used and structural and functional modifications may be made.

FIG. 1A illustrates an example of a suitable gaming system environment 100 on which computer games, video games, and/or other electronic games (collectively referred to herein as computer games) may be played. The gaming system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the features described herein. Neither should the gaming system environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the illustrative operating gaming system environment 100.

Aspects described herein are operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/ or configurations that may be suitable for use include, but are not limited to, personal computers; server computers; portable and hand-held devices such as personal digital assistants (PDAs), tablet PCs or laptop PCs; multiprocessor systems; microprocessor-based systems; set top boxes; programmable consumer electronics; network PCs; minicomputers; mainframe computers; electronic game consoles, distributed computing environments that include any of the above systems or devices; and the like.

Aspects herein may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The features described herein may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

FIG. 1 shows an exemplary gaming system 100. Gaming system 100 may include a game console 102 and one or more handheld controllers, as represented by controllers 104(1) and 104(2). The game console 102 may be equipped with an internal or external hard disk drive and a portable media drive 106 that supports various forms of portable storage media as represented by optical storage disc 108. Examples of suitable portable storage media include DVD, CD-ROM, game discs, and so forth.

Game console 102 may have a number of slots 110 on its front face to support up to four controllers, although the number and arrangement of slots may be modified. A power button 112 and an eject button 114 are also positioned on the front face of the game console 102. The power button 112 switches power to the game console and the eject button 114 alternately opens and closes a tray of the portable media drive 106 to allow insertion and extraction of the storage disc 108. In some aspects, game console 102 may be a dedicated computing device for home entertainment, and may be a closed, secure system that only executes authenticated and authorized applications. The game console 102 may be optimized for executing game programs (e.g., having increased processing support for gaming applications, such as physics co-processors, math co-processors, graphics co-processors, higher resolution video output, higher fidelity audio output, etc.), and may omit certain features commonly found on personal computing devices, such as an alphanumeric keyboard, internal hardware expansion slots, printer communication port, etc.

Game console 102 may connect to a television or other display (not shown) via A/V interfacing cables 120. A power cable 122 provides power to the game console. Game console 102 may further be configured with broadband network capabilities, as represented by the cable or modem connector 124 to facilitate access to a network, such as the Internet. Connector 124 may also be fitted with a wireless adapter to connect to one or more wireless networks.

Each controller 104 may be coupled to the game console 102 via a wire or wireless interface. In the illustrated implementation, the controllers are USB (Universal Serial Bus) compatible and are connected to the console 102 via USB cables 130. Controller 102 may be equipped with any of a wide variety of user interaction mechanisms. As illustrated in FIG. 1, each controller 104 may be equipped with two thumbsticks 132(1) and 132(2), a D-pad 134, buttons 136 (e.g., 'A', 'B', 'X', 'Y'), and two triggers 138.
thumbs 132 may be analog directional control units, and may include analog potentiometers to detect a degree of position in the X- and Y-coordinates. D-pad 134 may be a directional pad, with inputs for entering directional commands such as up, down, left and right, or combinations of these directions (e.g., upper-left). D-pad 134 may also be analog, and may provide input as to a degree of pressure used to press in a particular direction. These mechanisms are merely representative, and other known gaming mechanisms may be substituted for or added to those shown in FIG. 1.

[0036] A memory unit (MU) 140 may be inserted into the controller 104 to provide additional and portable storage. Portable memory units enable users to store game parameters and user accounts, and port them for play on other consoles. In the described implementation, each controller is configured to accommodate two memory units 140, although more or less than two units may be employed in other implementations. A headset 142 may be connected to the controller 104 or game console 102 to provide audio communication capabilities. Headset 142 may include a microphone for audio input and one or more speakers for audio output.

[0037] Gaming system 100 is capable of playing, for example, games, music, and videos. With the different storage offerings, titles can be played from the hard disk drive or the portable medium 108 in drive 106, from an online source, or from a memory unit 140. For security, in some embodiments executable code can only be run from the portable medium 108. A sample of what gaming system 100 is capable of playing include game titles played from CD and DVD discs, from the hard disk drive, or from an online source; digital music played from the CD in the portable media drive 106, from a file on the hard disk drive (e.g., “WINDSWTM” Media Audio (WMA) format), or from online streaming sources; and digital audio/video played from a DVD disc in the portable media drive 106, from a file on the hard disk drive (e.g., Active Streaming Format), or from online streaming sources.

[0038] FIG. 2 shows functional components of the gaming system 100 in more detail. The game console 102 has a central processing unit (CPU) 200 and a memory controller 202 that facilitates processor access to various types of memory, including a flash ROM (Read Only Memory) 204, a RAM (Random Access Memory) 206, a hard disk drive 208, and the portable media drive 106. The CPU 200 is equipped with a level 1 cache 210 and a level 2 cache 212 to temporarily store data and hence reduce the number of memory access cycles, thereby improving processing speed and throughput.

[0039] The CPU 200, memory controller 202, and various memory devices are interconnected via one or more buses, including serial and parallel buses, a memory bus, a peripheral bus, and a processor or local bus using any of a variety of bus architectures. By way of example, such architectures can include an Industry Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA) local bus, and a Peripheral Component Interconnects (PCI) bus also known as a Mezzanine bus.

[0040] As one suitable implementation, the CPU 200, memory controller 202, ROM 204, and RAM 206 are integrated onto a common module 214. In this implementation, ROM 204 is configured as a flash ROM that is connected to the memory controller 202 and a ROM bus (not shown). RAM 206 is configured as multiple DDR SDRAM (Double Data Rate Synchronous Dynamic RAM) that are independently controlled by the memory controller 202 via separate buses (not shown). The hard disk drive 208 and portable media drive 106 are connected to the memory controller via the PCI bus and an ATA (AT Attachment) bus 216.

[0041] A 3D graphics processing unit 220 and a video encoder 222 form a video processing pipeline for high speed and high resolution graphics processing. Data is carried from the graphics processing unit 220 to the video encoder 222 via a digital video bus (not shown). An audio processing unit 224 and an audio codec (coder/decoder) 226 form a corresponding audio processing pipeline with high fidelity and stereo processing. Audio data is carried between the audio processing unit 224 and the audio codec 226 via a communication link (not shown). The video and audio processing pipelines output data to an A/V (audio/video) port 228 for transmission to the television or other display. In the illustrated implementation, the video and audio processing components 220-228 are mounted on the module 214.

[0042] Also implemented on the module 214 are a USB host controller 230 and a network interface 232. The USB host controller 230 is coupled to the CPU 200 and the memory controller 202 via a bus (e.g., PCI bus) and serves as host for the peripheral controllers 104(1)-104(4). The network interface 232 provides access to a network (e.g., Internet, home network, etc.) and may be any of a wide variety of various wire or wireless interface components including an Ethernet card, a modem, a Bluetooth module, a cable modem, and the like.

[0043] The game console 102 has two dual controller support subassemblies 240(1) and 240(2), with each subassembly supporting two game controllers 104(1)-104(4). A front panel I/O subassembly 242 supports the functionality of the power button 112 and the eject button 114, as well as any LEDs (light emitting diodes) or other indicators exposed on the outer surface of the game console. The subassemblies 240(1), 240(2), and 242 are coupled to the module 214 via one or more cable assemblies 244. In some embodiments, game console 102 may also include a keyboard input subassembly 240(3), to which is connected a keyboard 254. The keyboard 254 and its subassembly might be offered as part of a developer’s kit version of the console 102, to allow the use of a keyboard for entering text commands and data for testing purposes. In some embodiments, the keyboard 254 may communicate directly with a controller port (e.g., as in subassemblies 240), and the use of a separate keyboard input subassembly is not necessary. Furthermore, to conserve further game console resources, a keyboard driver and subassembly may be omitted from the console, and instead the console may be coupled to a second computing device (e.g., another PC, or a debugging workstation) via USB cable, by which the second computing device may send command sequences to the game console, reducing the need in the game console for separate software and/or hardware for interpreting text command sequences entered via the keyboard.

[0044] Eight memory units 140(1)-140(8) are illustrated as being connectable to the four controllers 104(1)-104(4),
i.e., two memory units for each controller. Each memory unit 140 offers additional storage on which games, game parameters, and other data may be stored. When inserted into a controller, the memory unit 140 can be accessed by the memory controller 202.

[0045] A system power supply module 250 provides power to the components of the gaming system 100. A fan 252 cools the circuitry within the game console 102.

[0046] The game console 102 implements a uniform media portal model that provides a consistent user interface and navigation hierarchy to move users through various entertainment areas. The portal model offers a convenient way to access content from multiple different media types—game data, audio data, and video data—regardless of the media type inserted into the portable media drive 106.

[0047] To implement the uniform media portal model, a console user interface (UI) application 260 is stored on the hard disk drive 208. When the game console is powered on, various portions of the console application 260 are loaded into RAM 206 and/or caches 210, 212 and executed on the CPU 200. The console application 260 presents a graphical user interface that provides a consistent user experience when navigating to different media types available on the game console.

[0048] The gaming system 100 may be operated as a standalone system by simply connecting the system to a television or other display. In this standalone mode, the gaming system 100 allows one or more players to play games, watch movies, or listen to music. However, with the integration of broadband connectivity made available through the network interface 232, the gaming system 100 may further be operated as a participant in a larger network gaming community. This network gaming environment is described next.

[0049] FIG. 3 shows an exemplary network gaming environment 300 that interconnects multiple gaming systems 100(1), . . . , 100(g) via a network 302. The network 302 represents any of a wide variety of data communications networks. It may include public portions (e.g., the Internet) as well as private portions (e.g., a residential Local Area Network (LAN)), as well as combinations of public and private portions. Network 302 may be implemented using any one or more of a wide variety of conventional communications media including both wired and wireless media. Any of a wide variety of communications protocols can be used to communicate data via network 302, including both public and proprietary protocols. Examples of such protocols include TCP/IP, IPX/SPX, NetBEUI, etc.

[0050] In addition to gaming systems 100, one or more online services 304(1), . . . , 304(n) may be accessible via the network 302 to provide various services for the participants, such as hosting online games, serving downloadable music or video files, hosting gaming competitions, serving streaming audio/video files, and the like. The network gaming environment 300 may further involve a key distribution center 306 that plays a role in authenticating individual players and/or gaming systems 100 to one another as well as online services 304. The distribution center 306 distributes keys and service tickets to valid participants that may then be used to form games amongst multiple players or to purchase services from the online services 304.

[0051] The network gaming environment 300 introduces another memory source available to individual gaming systems 100—online storage. In addition to the portable storage medium 108, the hard disk drive 208, and the memory unit(s) 140, the gaming system 100(1) can also access data files available at remote storage locations via the network 302, as exemplified by remote storage 308 at online service 304(s).

[0052] FIG. 4 is a block diagram of another illustrative online gaming environment 400, e.g., “XBOX™ LIVE™” by Microsoft Corporation of Redmond, Wash. Multiple game consoles 402(1), 402(2), . . . , 402(n) are coupled to a security gateway 404 via a network 406. Each game console 402 can be, for example, a game console 102 of FIG. 1 or FIG. 2. Network 406 represents any one or more of a variety of conventional data communications networks. Network 406 will typically include packet switched networks, but may also include circuit switched networks. Network 406 can include wire and/or wireless portions. In one exemplary implementation, network 406 includes the Internet and may optionally include one or more local area networks (LANs) and/or wide area networks (WANs). At least a part of network 406 is a public network, which refers to a network that is publicly-accessible. Virtually anyone can access the public network.

[0053] In some situations, network 406 includes a LAN (e.g., a home network), with a routing device situated between game console 402 and security gateway 404. This routing device may perform network address translation (NAT), allowing the multiple devices on the LAN to share the same IP address on the Internet, and also operating as a firewall to protect the device(s) on the LAN from access by malicious or mischievous users via the Internet.

[0054] Security gateway 404 operates as a gateway between public network 406 and a private network 408. Private network 408 can be any of a wide variety of conventional networks, such as a local area network. Private network 408, as well as other devices discussed in more detail below, is within a data center 410 that operates as a secure zone. Data center 410 is made up of trusted devices communicating via trusted communications. Thus, encryption and authentication within secure zone 410 is not necessary. The private nature of network 408 refers to the restricted accessibility of network 408—access to network 408 is restricted to only certain individuals (e.g., restricted by the owner or operator of data center 410).

[0055] Security gateway 404 is a cluster of one or more security gateway computing devices. These security gateway computing devices collectively implement security gateway 404. Security gateway 404 may optionally include one or more conventional load balancing devices that operate to direct requests to be handled by the security gateway computing devices to appropriate ones of those computing devices. This directing or load balancing is performed in a manner that attempts to balance the load on the various security gateway computing devices approximately equally (or alternatively in accordance with some other criteria).

[0056] Also within data center 410 are: one or more monitoring servers 412; one or more presence and notification front doors 414, one or more presence servers 416, one or more notification servers 418, and a profile store 428 (collectively implementing a presence and notification ser-
vice or system 430); one or more match front doors 420 and one or more match servers 422 (collectively implementing a match service); and one or more statistics front doors 424 and one or more statistics servers 426 (collectively implementing a statistics service). The servers 416, 418, 422, and 426 provide services to game consoles 402, and thus can be referred to as service devices. Other service devices may also be included in addition to, and/or in place of, one or more of the servers 416, 418, 422, and 426. Additionally, although only one data center is shown in FIG. 4, alternatively multiple data centers may exist with which game consoles 402 can communicate. These data centers may operate independently, or alternatively may operate collectively (e.g., to make one large data center available to game consoles 102, 402).

[0057] Game consoles 402 are situated remotely from data center 410, and access data center 410 via network 406. A game console 402 desiring to communicate with one or more devices in the data center logs in to the data center and establishes a secure communication channel between the console 402 and security gateway 404. Game console 402 and security gateway 404 encrypt and authenticate data packets being passed back and forth, thereby allowing the data packets to be securely transmitted between them without being understood by any other device that may capture or copy the data packets without breaking the encryption. Each data packet communicated from game console 402 to security gateway 404, or from security gateway 404 to game console 402 can have data embedded therein. This embedded data is referred to as the content or data content of the packet. Additional information may also be inherently included in the packet based on the packet type (e.g., a heartbeat packet).

[0058] The secure communication channel between a console 402 and security gateway 404 is based on a security ticket. Console 402 authenticates itself and the current user(s) of console 402 to a key distribution center 420 and obtains, from key distribution center 420, a security ticket. Console 402 then uses this security ticket to establish the secure communication channel with security gateway 404. In establishing the secure communication channel with security gateway 404, the game console 402 and security gateway 404 authenticate themselves to one another and establish a session security key that is known only to that particular game console 402 and the security gateway 404. This session security key is used to encrypt data transferred between the game console 402 and the security gateway cluster 404, so no other devices (including other game consoles 402) can read the data. The session security key is also used to authenticate a data packet as being from the security gateway 404 or game console 402 that the data packet alleges to be from. Thus, using such session security keys, secure communication channels can be established between the security gateway 404 and the various game consoles 402.

[0059] Once the secure communication channel is established between a game console 402 and the security gateway 404, encrypted data packets can be securely transmitted between the two. When the game console 402 desires to send data to a particular service device in data center 410, the game console 402 encrypts the data and sends it to security gateway 404 requesting that it be forwarded to the particular service device(s) targeted by the data packet. Security gateway 404 receives the data packet and, after authenticating and decrypting the data packet, encapsulates the data content of the packet into another message to be sent to the appropriate service via private network 408. Security gateway 404 determines the appropriate service for the message based on the requested service(s) targeted by the data packet.

[0060] Similarly, when a service device in data center 410 desires to communicate data to a game console 402, the data center sends a message to security gateway 404, via private network 408, including the data content to be sent to the game console 402 as well as an indication of the particular game console 402 to which the data content is to be sent. Security gateway 404 embeds the data content into a data packet, and then encrypts the data packet so it can only be decrypted by the particular game console 402 and also authenticates the data packet as being from the security gateway 404.

[0061] Although discussed herein as primarily communicating encrypted data packets between security gateway 404 and a game console 402, alternatively some data packets may be partially encrypted (some portions of the data packets are encrypted while other portions are not encrypted). Which portions of the data packets are encrypted and which are not can vary based on the desires of the designers of data center 410 and/or game consoles 402. For example, the designers may choose to allow voice data to be communicated among consoles 402 so that users of the consoles 402 can talk to one another—the designers may further choose to allow the voice data to be unencrypted while any other data in the packets is encrypted. Additionally, in another alternative, some data packets may have no portions that are encrypted (that is, the entire data packet is unencrypted). It should be noted that, even if a data packet is unencrypted or only partially encrypted, all of the data packet can still be authenticated.

[0062] Each security gateway device in security gateway 404 is responsible for the secure communication channel with typically one or more game consoles 402, and thus each security gateway device can be viewed as being responsible for managing or handling one or more game consoles. The various security gateway devices may be in communication with each other and communicate messages to one another. For example, a security gateway device that needs to send a data packet to a game console that it is not responsible for managing may send a message to all the other security gateway devices with the data to be sent to that game console. This message is received by the security gateway device that is responsible for managing that game console and sends the appropriate data to that game console. Alternatively, the security gateway devices may be aware of which game consoles are being handled by which security gateway devices—this may be explicit, such as each security gateway device maintaining a table of game consoles handled by the other security gateway devices, or alternatively implicit, such as determining which security gateway device is responsible for a particular game console based on an identifier of the game console.

[0063] Monitoring server(s) 412 operate to inform devices in data center 410 of an unavailable game console 402 or an unavailable security gateway device of security gateway 404. Game consoles 402 can become unavailable for a variety of different reasons, such as a hardware or software
failure, the console being powered-down without logging out of data center 410, the network connection cable to console 402 being disconnected from console 402, other network problems (e.g., the LAN that the console 402 is on malfunctioning), etc. Similarly, a security gateway device of security gateway 404 can become unavailable for a variety of different reasons, such as hardware or software failure, the device being powered-down, the network connection cable to the device being disconnected from the device, other network problems, etc.

[0064] Each of the security gateway devices in security gateway 404 is monitored by one or more monitoring servers 412, which detect when one of the security gateway devices becomes unavailable. In the event a security gateway device becomes unavailable, monitoring server 412 sends a message to each of the other devices in data center 410 (servers, front doors, etc.) that the security gateway device is no longer available. Each of the other devices can operate based on this information as it sees fit (e.g., it may assume that particular game consoles being managed by the security gateway device are no longer in communication with data center 410 and perform various clean-up operations accordingly). Alternatively, only certain devices may receive such a message from the monitoring server 412 (e.g., only those devices that are concerned with whether security gateway devices are available).

[0065] Security gateway 404 monitors the individual game consoles 402 and detects when one of the game consoles 402 becomes unavailable. When security gateway 404 detects that a game console is no longer available, security gateway 404 sends a message to monitoring server 412 identifying the unavailable game console. In response, monitoring server 412 sends a message to each of the other devices in data center 410 (or alternatively only selected devices) that the game console is no longer available. Each of the other devices can then operate based on this information as it sees fit.

[0066] Presence server(s) 416 hold and process data concerning the status or presence of a given user logged in to data center 410 for online gaming. Notification server(s) 418 maintains multiple notification queues of outgoing messages destined for a player logged in to data center 410. Presence and notification front door 414 is one or more server devices that operate as an intermediary between security gateway 404 and servers 416 and 418. One or more load balancing devices (not shown) may be included in presence and notification front door 414 to balance the load among the multiple server devices operating as front door 414. Security gateway 404 communicates messages for servers 416 and 418 to the front door 414, and the front door 414 identifies which particular server 416 or particular server 418 the message is to be communicated to. By using front door 414, the actual implementation of servers 416 and 418, such as which servers are responsible for managing data regarding which users, is abstracted from security gateway 404. Security gateway 404 can simply forward messages that target the presence and notification service to presence and notification front door 414 and rely on front door 414 to route the messages to the appropriate one of server(s) 416 and server(s) 418.

[0067] Match server(s) 422 hold and process data concerning the matching of online players to one another. An online user is able to advertise a game available for play along with various characteristics of the game (e.g., the location where a football game will be played, whether a game is to be played during the day or at night, the user’s skill level, etc.). These various characteristics can then be used as a basis to match up different online users to play games together. Match front door 420 includes one or more server devices (and optionally a load balancing device(s)) and operates to abstract match server(s) 422 from security gateway 404 in a manner analogous to front door 414 abstracting server(s) 416 and server(s) 418.

[0068] Statistics server(s) 426 hold and process data concerning various statistics for online games. The specific statistics used can vary based on the game designer’s desires (e.g., the top ten scores or times, a world ranking for all online players of the game, a list of users who have found the most items or spent the most time playing, etc.). Statistics from front door 426 includes one or more server devices (and optionally a load balancing device(s)) and operates to abstract statistics server(s) 426 from security gateway 404 in a manner analogous to front door 414 abstracting server(s) 416 and server(s) 418.

[0069] Title server(s) 432 may be provided for each individual game title or program supported by the online service. Each title server 432 may act as a server and communicate with game consoles 402 to execute programs and/or modules for conducting an online gaming session. The server 432 may be given as much, or as little, control over actions in a game as desired by the game developer. For example, some games may rely on the centralized title server 432 to perform calculations and processing having to do with interactions among multiple players on different game consoles 402, such as determining which player character was first to a particular point on a map, or controlling the appearance of a common environment in which the various players’ characters appear, and may rely on the individual consoles 402 to handle the just the more localized aspects, such as the actual rendering of images and results, and initial processing of user inputs.

[0070] On the other hand, games may rely more on the consoles 402 for processing. Processing responsibilities for a given game may be distributed among a plurality of consoles 402 that are connected to a game session. Or, the game may select a subset of the consoles 402 connected to the session, and cause that console to act as the server for the session. In such a situation, title server 432 may simply coordinate communications between the game consoles 402, or may simply perform initial startup functions (e.g., game type, startup authorization, etc.), and then await results.

[0071] Tournament servers 434 may also be present to manage tournament functionality. This functionality will be described in greater detail below, and may generally relate to the creation, management and/or termination of online multiplayer tournaments. The various servers described above need not reside on separate machines, as some or all of them may be combined into a common process or machine.

[0072] Thus, it can be seen that security gateway 404 operates to shield devices in the secure zone of data center 410 from the untrusted, public network 406. Communications within the secure zone of data center 410 need not be encrypted, as all devices within data center 410 are trusted. However, any information to be communicated from a
device within data center 410 to a game console 402 passes through security gateway cluster 404, where it is encrypted in such a manner that it can be decrypted by only the game console 402 targeted by the information.

[0073] As illustrated in FIG. 4, title server 432 and tournament server 434 may reside behind a second security gateway 405. Security gateway 405 may be the same as security gateway 404, or it may be a simplified one to omit features not needed by its servers and devices.

[0074] One or more features described herein may be embodied in computer-executable instructions (i.e., software) stored in RAM memory 206, non-volatile memory 108, 208, 308, or any other resident memory on game console 102. Generally, software modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types when executed by a processor in a computer or other device. The computer executable instructions may be stored on a computer readable medium such as one or more hard disks 208, removable storage media 108 (e.g., CD-ROM, DVD, disk, etc.), solid state memory, RAM 206, etc. As will be appreciated by one of skill in the art, the functionality of the software modules may be combined or distributed as desired in various embodiments. In addition, the functionality may be embodied in whole or in part in firmware or hardware equivalents such as application specific integrated circuits (ASIC), field programmable gate arrays (FPGA), and the like.

[0075] Aspects herein are not limited to console computing environments. Indeed, these aspects may also be implemented in video games that operate on personal computers (PC). FIG. 5A illustrates an example of a suitable computing system environment 500 on which the features described herein may be implemented. The computing system environment 500 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the features described herein. Neither should the computing environment 500 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 500.

[0076] The features herein are operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0077] The features herein may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc., that perform particular tasks or implement particular abstract data types. The features may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[0078] With reference to FIG. 5A, an exemplary system for implementing the features described herein includes a general purpose computing device in the form of a computer 510. Components of computer 510 may include, but are not limited to, a processing unit 520, a system memory 530, and a system bus 521 that couples various system components including the system memory to the processing unit 520. The system bus 521 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0079] Computer 510 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 510 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, and removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices (in the singular or the plural), or any other medium which can be used to store the desired information and which can be accessed by computer 510. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term “modulated data signal” means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media.

[0080] The system memory 530 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 531 and random access memory (RAM) 532. A basic input/output system 533 (BIOS), containing the basic routines that help to transfer information between elements within computer 510, such as during start-up, is typically stored in ROM 531. RAM 532 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 520. By way of example, and not limitation, FIG. 5A illustrates operating system 534, application programs 535, other program modules 536, and program data 537.
The computer 510 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 5 illustrates a hard disk drive 541 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 551 that reads from or writes to a removable, nonvolatile magnetic disk 552, and an optical disk drive 555 that reads from or writes to a removable, nonvolatile optical disk 556 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 541 is typically connected to the system bus 521 through a non-removable memory interface such as interface 540, and magnetic disk drive 551 and optical disk drive 555 are typically connected to the system bus 521 by a removable memory interface, such as interface 550.

The drives and their associated computer storage media discussed above and illustrated in FIG. 5A, provide storage of computer readable instructions, data structures, program modules and other data for the computer 510. In FIG. 5, for example, hard disk drive 541 is illustrated as storing operating system 544, application programs 545, other program modules 546, and program data 547. Note that these components can either be the same as or different from operating system 534, application programs 535, other program modules 536, and program data 537. Operating system 544, application programs 545, other program modules 546, and program data 547 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer through input devices such as a keyboard 562 and pointing device 561, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 520 through a user input interface 560 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 591 or other type of display device is also connected to the system bus 521 via an interface, such as a video interface 590. In addition to the monitor, computers may also include other peripheral output devices such as speakers 597 and printer 596, which may be connected through an output peripheral interface 595.

When used in a LAN networking environment, the computer 510 is connected to the LAN 571 through a network interface or adapter 570. When used in a WAN networking environment, the computer 510 typically includes a modem 572 or other means for establishing communications over the WAN 573, such as the Internet. The modem 572, which may be internal or external, may be connected to the system bus 521 via the user input interface 560, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 510, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 5 illustrates remote application programs 585 as residing on memory device 581. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

A programming interface (or more simply, interface) may be viewed as any mechanism, process, protocol for enabling one or more segment(s) of code to communicate with or access the functionality provided by one or more other segment(s) of code. Alternatively, a programming interface may be viewed as one or more mechanism(s), method(s), function call(s), module(s), object(s), etc. of a component of a system capable of communicative coupling to one or more mechanism(s), method(s), function call(s), module(s), etc. of other component(s). The term “segment of code” in the preceding sentence is intended to include one or more instructions or lines of code, and includes, e.g., code modules, objects, subroutines, functions, and so on, regardless of the terminology applied or whether the code segments are separately compiled, or whether the code segments are provided as source, intermediate, or object code, whether the code segments are utilized in a runtime system or process, or whether they are located on the same or different machines or distributed across multiple machines, or whether the functionality represented by the segments of code are implemented wholly in software, wholly in hardware, or a combination of hardware and software.

Notionally, a programming interface may be viewed generically, as shown in FIG. 5B or FIG. 5C. FIG. 5B illustrates an interface Interface1 as a conduit through which first and second code segments communicate. FIG. 5C illustrates an interface as comprising interface objects 11 and 12 (which may or may not be part of the first and second code segments), which enable first and second code segments of a system to communicate via medium M. In the view of FIG. 5C, one may consider interface objects 11 and 12 as separate interfaces of the same system and one may also consider that objects 11 and 12 plus medium M comprise the interface. Although FIGS. 5B and 5C show bi-directional flow and interfaces on each side of the flow, certain implementations may only have information flow in one direction (or no information flow as described below) or may only have an interface object on one side. By way of example, and not limitation, terms such as application programming interface (API), entry point, method, function, subroutine, remote procedure call, and component object model (COM) interface, are encompassed within the definition of programming interface.

Aspects of such a programming interface may include the method whereby the first code segment transmits information (where “information” is used in its broadest
sense and includes data, commands, requests, etc.) to the second code segment; the method whereby the second code segment receives the information; and the structure, sequence, syntax, organization, schema, timing and content of the information. In this regard, the underlying transport medium itself may be unimportant to the operation of the interface, whether the medium be wired or wireless, or a combination of both, as long as the information is transported in the manner defined by the interface. In certain situations, information may not be passed in one or both directions in the conventional sense, as the information transfer may be either via another mechanism (e.g., information placed in a buffer, file, etc. separate from information flow between the code segments) or non-existent, as when one code segment simply accesses functionality performed by a second code segment. Any or all of these aspects may be important in a given situation, e.g., depending on whether the code segments are part of a system in a loosely coupled or tightly coupled configuration, and so this list should be considered illustrative and non-limiting.

[0088] This notion of a programming interface is known to those skilled in the art and is clear from the foregoing detailed description of the invention. There are, however, other ways to implement a programming interface, and, unless expressly excluded, these are intended to be encompassed by the claims set forth at the end of this specification. Such other ways may appear to be more sophisticated or complex than the simplistic view of Figs. 5B and 5C, but they nonetheless perform a similar function to accomplish the same overall result. We will now briefly describe some illustrative alternative implementations of a programming interface.

A. Factorizing

[0089] A communication from one code segment to another may be accomplished indirectly by breaking the communication into multiple discrete communications. This is depicted schematically in Figs. 5D and 5E. As shown, some interfaces can be described in terms of divisible sets of functionality. Thus, the interface functionality of Figs. 5B and 5C may be factorized to achieve the same result, just as one may mathematically provide 24, or 2 times 3 times 2. Accordingly, as illustrated in Fig. 5D, the function provided by interface Interface1 may be subdivided to convert the communications of the interface into multiple interfaces Interface1A, Interface1B, Interface1C, etc. while achieving the same result. As illustrated in Fig. 5E, the function provided by interface Interface1 may be subdivided into multiple interfaces Interface 1a, Interface 1b, Interface 1c, etc. while achieving the same result. Similarly, interface Interface 2 of the second code segment which receives information from the first code segment may be factorized into multiple interfaces Interface 2a, Interface 2b, Interface 2c, etc. When factorizing, the number of interfaces included with the 1st code segment need not match the number of interfaces included with the 2nd code segment. In either of the cases of Figs. 5D and 5E, the functional spirit of interfaces Interface1 and Interface2 remain the same as with Figs. 5B and 5C, respectively. The factorizing of interfaces may also follow associative, commutative, and other mathematical properties such that the factorizing may be difficult to recognize. For instance, ordering of operations may be unimportant, and consequently, a function carried out by an interface may be carried out well in advance of reaching the interface, by another piece of code or interface, or performed by a separate component of the system. Moreover, one of ordinary skill in the programming arts can appreciate that there are a variety of ways of making different function calls that achieve the same result.

B. Redefinition

[0090] In some cases, it may be possible to ignore, add or redefine certain aspects (e.g., parameters) of a programming interface while still accomplishing the intended result. This is illustrated in Figs. 5F and 5G. For example, assume interface Interface1 of Fig. 5B includes a function call Square (input, precision, output), a call that includes three parameters, input, precision and output, and which is issued from the 1st Code Segment to the 2nd Code Segment. If the middle parameter precision is of no concern in a given scenario, as shown in Fig. 5F, it could just as well be ignored or even replaced with a meaningless (in this situation) parameter. One may also add an additional parameter of no concern. In either event, the functionality of square can be achieved, since output is returned after input is squared by the second code segment. Precision may very well be a meaningful parameter to some downstream or other portion of the computing system; however, once it is recognized that precision is not necessary for the narrow purpose of calculating the square, it may be replaced or ignored. For example, instead of passing a valid precision value, a meaningless value such as a birth date could be passed without adversely affecting the result. Similarly, as shown in Fig. 5G, interface Interface1 is replaced by interface Interface1', redefined to ignore or add parameters to the interface. Interface Interface1' may similarly be redefined as Interface1'2, redefined to ignore unnecessary parameters, or parameters that may be processed elsewhere. The point here is that in some cases a programming interface may include aspects, such as parameters, which are not needed for some purpose, and so they may be ignored or redefined, or processed elsewhere for other purposes.

C. Inline Coding

[0091] It may also be feasible to merge some or all of the functionality of two separate code modules such that the "interface" between them changes form. For example, the functionality of Figs. 5B and 5C may be converted to the functionality of Figs. 5H and 5I, respectively. In Fig. 5H, the previous 1st and 2nd Code Segments of Fig. 5B are merged into a module containing both of them. In this case, the code segments may still be communicating with each other, but the interface may be adapted to a form which is more suitable to the single module. Thus, for example, formal Call and Return statements may no longer be necessary, but similar processing or response(s) pursuant to interface Interface1 may still be in effect. Similarly, shown in Fig. 5I, part (or all) of interface Interface12 from Fig. 1C may be written inline into interface Interface11 to form interface Interface11'. As illustrated, interface Interface12 is divided into Interface12a and Interface12b, and interface portion Interface12a has been coded in-line with interface Interface11 to form interface Interface11'. For a concrete example, consider that the interface Interface11 from Fig. 5C performs a function call square (input, output), which is received by interface Interface12, which after processing the value passed with input (to square it) by the second code segment, passes back the squared result with output. In such a case, the processing performed by the second code segment (squaring input) can be performed by the first code segment without a call to the interface.
D. Divorce

[0092] A communication from one code segment to another may be accomplished indirectly by breaking the communication into multiple discrete communications. This is depicted schematically in FIGS. 5J and 5K. As shown in FIG. 5I, one or more piece(s) of middleware (Divorce Interface(s)), since they divorce functionality and/or interface functions from the original interface, are provided to convert the communications on the first interface, Interface1, to conform them to a different interface, in this case interfaces Interface2A, Interface2B and Interface2C. This might be done, e.g., where there is an installed base of applications designed to communicate with, say, an operating system in accordance with an Interface1 protocol, but then the operating system is changed to use a different interface, in this case interfaces Interface2A, Interface2B and Interface2C. The point is that the original interface used by the 2nd Code Segment is changed such that it is no longer compatible with the interface used by the 1st Code Segment, and so an intermediary is used to make the old and new interfaces compatible. Similarly, as shown in FIG. 5K, a third code segment can be introduced with divorce interface D11 to receive the communications from interface 11 and with divorce interface D12 to transmit the interface functionality to, for example, interfaces 12a and 12b, redesigned to work with D12, but to provide the same functional result. Similarly, D11 and D12 may work together to translate the functionality of interfaces 11 and 12 of FIG. 5C to a new operating system, while providing the same or similar functional result.

E. Rewriting

[0093] Yet another possible variant is to dynamically rewrite the code to replace the interface functionality with something else but which achieves the same overall result. For example, there may be a system in which a code segment presented in an intermediate language (e.g., Microsoft IL, Java ByteCode, etc.) is provided to a Just-in-Time (JIT) compiler or interpreter in an execution environment (such as that provided by the .Net framework, the Java runtime environment, or other similar runtime type environments). The JIT compiler may be written so as to dynamically convert the communications from the 1st Code Segment to the 2nd Code Segment, i.e., to conform them to a different interface as may be required by the 2nd Code Segment (either the original or a different 2nd Code Segment). This is depicted in FIGS. 5L and 5M. As can be seen in FIG. 5L, this approach is similar to the Divorce scenario described above. It might be done, e.g., where an installed base of applications are designed to communicate with an operating system in accordance with an Interface1 protocol, but then the operating system is changed to use a different interface. The JIT Compiler could be used to conform the communications on the fly from the installed-base applications to the new interface of the operating system. As depicted in FIG. 5M, this approach of dynamically rewriting the interface(s) may be applied to dynamically factor, or otherwise alter the interface(s) as well.

[0094] It is also noted that the above-described scenarios for achieving the same or similar result as an interface via alternative embodiments may also be combined in various ways, serially and/or in parallel, or with other intervening code. Thus, the alternative embodiments presented above are not mutually exclusive and may be mixed, matched and combined to produce the same or equivalent scenarios to the generic scenarios presented in FIGS. 53 and 5C. It is also noted that, as with most programming constructs, there are other similar ways of achieving the same or similar functionality of an interface which may not be described herein, but nonetheless are represented by the spirit and scope of the invention, i.e., it is noted that it is at least partly the functionality represented by, and the advantageous results enabled by, an interface that underlie the value of an interface. The aforementioned systems may be used to implement the debugging methods or systems described herein. Various features provide for the automation of providing debugging feedback information to a server by dumping portions of memory and automatically sending the dumped memory to a server when certain debugging events occur.

[0095] FIG. 6 illustrates an example overall process by which a tournament may be conducted in an online multiplayer gaming environment. At step 601, a tournament administrator may take steps to log in to a secure server, such as tournament server 434, to begin the creation of an online tournament. The administrator may be a game developer, player, or anyone having the right to create new tournament types on the system. The log in process may involve any desired mode of access, such as accessing a secure Internet site using a password and a computer terminal 510, or accessing an online location using a video game user interface from a console 402. In step 602, the tournament administrator may define the various parameters for the tournament being created. This entry may be accomplished using a displayed interface, such as an Internet page or a video game user interface, where the administrator may be prompted to enter (or select from a list) the various parameters for the tournament being defined. Example parameters are discussed further below.

[0096] When the tournament parameters have been defined, the process may then proceed to step 603, where the system (e.g., the tournament server 434, or title server 432, depending on which server is designated for the task) awaits a trigger for instantiating a tournament. The trigger may be of any type of predetermined event, such as the passage of a predetermined amount of time (e.g., an hour, 30 minutes, etc.), or the registration by a predetermined number of players (e.g., 10, 50, 100 players, etc.).

[0097] When the trigger is received or occurs, the process may proceed with instantiation of a tournament in step 604. Instantiation may involve the execution of one or more processing threads that will carry out the tournament as specified in the parameters. These threads may be executed by the tournament server 434, title server 432, individual consoles 402, or any other computing component in the system, and may be distributed across components (e.g., across multiple consoles 402) if desired. An instance of a tournament may be assigned its own unique tournament ID to help coordinate communications with clients participating in the tournament instance.

[0098] When a tournament instance is created, the tournament may be assigned a leaderboard ID. The underlying online service (e.g., XBOX LIVE®) may provide a common set of leaderboard data structures that are available for use by the various game programs supported by the con-
soles. Alternatively, the leaderboard data structures may be a custom structure created to support each tournament instance by, for example, holding the qualification scores for the given tournament instance. These structures may be a defined database object storing identifications for various players, and one or more data values for a player’s performance in a game (e.g., a player’s race time, favorite car, score, win/loss record, total scores, opponents played, etc.), and game clients on consoles 402, tournament servers, and other devices may run SQL queries to the leaderboard data structure to obtain game session and tournament results. The leaderboard may be managed as a software process by a title server 432, a statistics server 424, or any other desired device in the system. In some instances, different leaderboard structures may be used for different aspects of a tournament. For example, a title server 432 may manage a qualification leaderboard, which accepts and process the qualification data for various tournament entrants (explained below), while a statistics server 424 may manage an overall scores leaderboard data structure to track player progress through the tournament.

[0099] Once the tournament is instantiated, the process conducts the tournament in step 605. The actual execution of the tournament code to manage the tournament can occur in any server, such as on the Tournament Server 434, Title Server 432, and/or game consoles 402, as different tournaments will be conducted differently depending on the parameters chosen by the tournament administrator. Example tournaments are described further below with respect to FIGS. 8-9.

[0100] When the tournament (or a portion thereof, such as a round) has been conducted, the process may proceed to step 606, where the various game clients report their results to the server managing the tournament. Clients may be game code running on individual consoles 402, and each console may be responsible for reporting results to the tournament’s server. Those results may include information regarding that particular console 402’s player’s performance, and may also include information regarding the performance of some or all of the other players that participated in the game session (e.g., a player’s console may report not only that player’s score, but the scores of all the other players in the game session as well—this multiple reported information can be compared against each other to confirm that no cheating has occurred, as further discussed below). Alternatively, since multiple consoles 402 may participate in a single game session, this reporting may be coordinated among the consoles 402 such that one of the consoles participating in the session may be designated to report the results for everyone who played in the session, and the other consoles in the session may avoid duplicating that reporting. These results may also be reported to the server running the leaderboard, such as Statistics Server 424, using the secure communications required by that server. The separate reporting of the results to the leaderboard may help resolve anomalies in data that is reported to the tournament server, as well be addressed in greater detail below.

[0101] The actual results may be in any form, depending on the type of game being played, and may generally report on accomplishments made by the players in the game session. For example, a first-person shooter type game may report the player’s hit ratio, favorite weapon, etc., while a football game may report the win/loss outcome of the game, the score, and other player and/or game statistics. Games may use their own criteria in evaluating a player’s performance and awarding points, rank, or other rewards, and such awards may be reported as the result.

[0102] Since the tournament’s server will be receiving tournament results from a variety of clients, and may receive multiple reports of each player’s performance, inconsistencies and anomalies may occur. For example, two consoles 402 may report inconsistent game outcomes for a single game, or for a single player in the game (e.g., each reporting a different winner, score, statistic, etc.). In games in which host migration occurs (e.g., games where one of the consoles participating in the session is designated as a local server host for the game, but where that console experiences a dropout or failure during the session and a different participating console takes over as host), different consoles may report incomplete or otherwise inconsistent results. In step 607, the tournament’s server may compare results reported from different clients, and look for inconsistencies in the reported results. Additionally, the server may compare the results with a predefined data structure identifying normal or acceptable results to see if any reported results are beyond the bounds of what is considered normal or acceptable for the game session that was played. For example, in a typical football game, the game designer may decide that a score exceeding a predefined point total (e.g., more than 100 points) may be abnormal.

[0103] If an anomaly is detected, the process moves to step 608, where an arbitration process analyzes the reported results to address and/or correct the anomaly. This arbitration process may involve querying the leaderboard server to retrieve individual player results, and performing the comparison. Or, the arbitration may involve reporting the apparent conflict to the server managing the leaderboard to which the tournament is assigned, and that server (e.g., the Statistics Server 424) may make its own comparison of the results it received, and may run additional security and/or integrity checks to confirm which set of results is the genuine set. The arbitration process may access stored data identifying the relative credibility of the players involved. When the arbitration process identifies the error in the reported results, the process in step 609 may update the leaderboard data and correct the error.

[0104] If no anomalies were detected, or after anomalies are corrected, the process may proceed to step 610 and check to see whether the tournament should be concluded. Different tournaments may have different parameters defining how the tournament will end. For example, a Tournament Winner parameter may be used as described further below. In this step, the tournament process may compare the reported results, or the total results, with the parameter(s) defining a tournament winner, and decide whether a winner has been found and the tournament should conclude.

[0105] If the tournament is not to be concluded, the process may return to step 605 to continue conducting the tournament. For example, this may entail processing the next round in the tournament.

[0106] If the tournament is to be concluded, the process may move to step 611 and conclude the tournament. This conclusion may involve the final tallying of scores and results from the leaderboard data object to determine player rankings and a tournament winner. Concluding a tournament
will often involve the awarding of a prize to the winner and those finishing in predefined places (e.g., 2nd, 3rd, top ten, etc.). The prizes may be physical items provided by a tournament sponsor, or in-game virtual items such as a race car with particular design, special equipment or weapon, etc. Concluding the tournament may also include releasing the leaderboard ID to allow the tournament’s leaderboard data object to be used for another tournament or another game, recording results in an archival database for future access, recording the high scores, etc. Upon conclusion of a tournament, the process may then return to step 603 to await the next triggering event. If desired, the tournament trigger parameter may be defined to automatically create the tournament instance if one does not already exist, or immediately after a tournament concludes, such that the tournament may be continuously available for players. As noted above, multiple instances of a tournament may exist simultaneously, if desired, so the instantiation of a second tournament need not depend on the conclusion of a prior instance. For example, the conclusion of a tournament instance may simply end the process, with separate processes occurring for each independent instance.

FIG. 6 above referred to a step of defining tournament parameters, and to elaborate on that step, the following discussion addresses several example tournament parameters that may be used:

Tournament Name—a text string given by the administrator to identify the tournament to participants (e.g., the “Halo 2™ Slayfast Sponsored by Microsoft”).

Tournament Owner—an identification of an administrator for the tournament. For example, the owner of a tournament may be the game developer, or a sponsor for the tournament.

Tournament Type—an identification of a predefined type of tournament. The Tournament Server 434 may offer a number of predefined types of tournaments (e.g., single elimination, double elimination, others as described below, etc.) that may be accessed and customized by the administrator.

Number of Tournament Entrants—one or more values identifying a minimum, maximum, range, and/or preferred number of total player participants supported in the tournament. Some tournaments may be better suited for smaller-scales, while other formats may be more ideal for larger numbers of players.

Number of Game Session Entrants—one or more values identifying a minimum, maximum, range, and/or preferred number of players participating in each game session of the tournament. A game session may be defined by each game differently, and may refer to the basic unit of play in the game. This may differ depending on the game type. For example, a game of chess involves two entrants per session (e.g., a board), but a game of poker might involve five entrants per session (e.g., a table).

Number of Rounds—one or more values identifying a minimum, maximum, range, and/or preferred number of rounds for the tournament. A round of a tournament may be defined to occur when each tournament entrant plays in one game session.

Window Factor—in some tournaments, players for a particular game session (e.g., a single “death match” involving a group of players) are grouped together based, at least in part, on how long it has been since the last time the players played against one another. The window factor may identify a time, which may be measured in terms of game sessions, rounds and/or tournaments, as well as seconds, minutes, hours, days, weeks, etc., that is required to pass before the players may play together again in the same game session of the tournament. For example, a tournament might specify a window factor of two game sessions, which means before two players will be grouped together for a game, they must not have played against one another for at least two games. The use of a window factor may be particularly desirable for tournaments that have unlimited duration, or very large numbers of matches and players. To support the use of a window factor, the tournament instance may store a data table identifying the last time each pair of players in the tournament played against one another in the tournament.

Rank Gap—in some tournaments, players may be grouped together according to their score, rank, skill level, etc., and a rank gap may identify a range of ranks within which players are permitted to be grouped together. For example, a tournament might wish to only group players together if they are within 20 points of one another, or if their overall rankings are within a number of ranks (e.g., players no more than 10 ranks higher or lower than one another) or percentage (e.g., percentage of the total number of entrants in the tournament) of one another.

Scheduling Options—these options identify the schedule in which the tournament will be played. A tournament may be continuous, in which a new round is begun automatically upon completion of a prior round, or according to a schedule. The schedule may indicate certain time frames in which rounds and game sessions may be conducted, such as a weekend-only tournament, a daily tournament, a specific date and/or time of day, every 30 minutes, etc.

New player factor—one or more values identifying how new players’ scores will be used in rankings. Point rankings are addressed in greater detail below, but for some tournaments, players are ranked according to data values that might not make a lot of statistical sense in the very beginning of the player’s participation. For example, a tournament that simply ranks players based on win/loss record might rank an undefeated 30-0 player the same as a new player with a 1-0 record (both are 100% winners), even though that new player might not truly deserve such a lofty rank. To accommodate for this, the tournament may use a new player factor, in which new player’s scores are adjusted before used in ranking. So, for example, a new player’s score might be reduced by 75% when only one game session’s score is available, 50% when two game sessions’ scores are available, 25% when three game sessions’ scores are available, and not discounted at all if four or more game sessions’ scores are available. The new player factor may identify a duration for the effect (e.g., number of rounds or game session), and an adjustment value for the effect (e.g., score discounting, point value addition/subtraction, etc.).

Trigger event—an identification of one or more events that will cause an instance of a tournament to be created. This may relate to the scheduling parameter described above, in which case a trigger event could simply
be the time of day matching the schedule. Other types of triggers may also be used. For example, a tournament might automatically begin when a predetermined number of players have registered for the tournament, or upon entry of a command by one or more players already registered for the tournament (e.g., all registered players pressing a controller button to signal their readiness).

**[0119]** Tournament winner—one or more values identifying the manner in which a tournament winner is determined. This may be a predetermined point total (e.g., the first player to reach 1000 points; or the person with the lowest total time in a race), a predetermined game event (e.g., the first player to accomplish a given objective within the game) or a ranking following a number of rounds (e.g., the player with the top ranking after the last round), etc.

**[0120]** Game-specific Options—a number of other options that may be specific to particular games. These may include the identification of race tracks, weather conditions, car types, and point scoring for a racing game tournament, or the identification of map type, weapon loadout, and scoring for a first-person shooter tournament.

**[0121]** The various tournament parameters may be stored in a tournament data package data structure 701, as shown in FIG. 7. This data structure may include one or more round listings 702, where each round listing includes data for a segment/stage/subset of rounds in a tournament. For example, the example tournament 701 is a tournament including thirteen rounds, divided into three segments or stages. The round list may include a header portion 703 containing information for the particular list, such as an identification of the number of rounds in the list, and common settings, parameters, or themes for the rounds in the list (e.g., all races in list 1 will be on the “New York” race track, or all races in list 1 will occur with the weather set to "rainy"). The list may also include a plurality of individual round settings 704, containing parameters that will be used for each corresponding game session. For example, a first round setting might indicate that only low-power cars may be used for the first race, while the second round setting might indicate that only four-wheel drive cars may be used for the second race. The following pseudocode provides an example of a tournament data package:

```
<GAMESETTINGS>
  <PACKAGE 1>
    <NAME> Text String Name of Package <NAME>
    <LIST 1>
      <NAME> Handle for List <NAME>
      <VISIBLENAME> Displayed Text String Name of List <VISIBLENAME>
      <ROUND>
        <CODE 1>1011<CODE 1>
        <CODE 2>21000<CODE 2>
        <CODE 3>31190<CODE 3>
        <CODE 4>40<CODE 4>
      </ROUND>
    </LIST 1>
  </PACKAGE 1>
</GAMESETTINGS>
```

In the preceding example, a tournament settings file may include multiple packages, each package may include multiple lists, and each list may include data values for the list’s internal handle, a text value to be displayed, and one or more game codes for settings in the game. The game codes may be numeric values that correspond to entries in a table used by the game program for configuring the game. For example, a racing game may be provided with sixteen different race tracks, and the code ‘1011’ may be a binary index into a table listing these tracks, identifying the 11th track for use. Other codes may similarly be indices into other settings tables, such as tables for the weather, the car transmission, the direction on the track, type of car, etc. The following example illustrates how such a table may be configured:

```
<TRANSMISSION TYPE>
  0-Manual
  1-Automatic
<WEATHER TYPE>
  0-Normal
  1-Sunny and Hot
  2-Rainy
  3-Snow
  4-Ice
  5-Heavy Wind
```

Additionally, the values need not all be index codes into tables. For example, a value for the maximum horsepower may simply include a numeric value for that maximum horsepower. Other data may also be included, such as the schedule data for the tournament instance (e.g., how frequently the tournament may instance, conditions, etc.), and for individual rounds within an instance (e.g., how often rounds occur).

**[0134]** As noted above, there may be a variety of tournament types. Single elimination, double elimination and round-robin type tournaments may be used. FIG. 8 illustrates an example of another type of tournament that can be used. In the FIG. 8 tournament, players begin in step 801 by
registering to participate in the tournament. Registration may be done in a variety of ways. For example, the player may log onto a server, such as Title Server 432 or Tournament Server 434, and query the server to obtain a list of available tournaments that are open for registration. The queried server may return a list of available tournaments, displaying the relevant tournament parameters (e.g., tournament name, number of rounds, number of entrants already registered, start time, prize, etc.), and the user may register for one by pressing a button on a controller with a particular tournament highlighted. Alternatively, players may join a tournament as a result of receiving an invitation sent by another player who has already registered for the tournament.

Player registration step 801 may continue until a predetermined time (e.g., elapsed time since tournament instance was created, time of day for scheduled tournament, elapsed time since first registrant, etc.) passes, or until a predetermined number of registrants have joined (e.g., the tournament parameter defining a maximum number of entrants). When registration ends, the tournament may begin by initially placing the entrants in a random, or pseudo-random, order in step 802. This initial placement of ranking can be done in any desired way, such as alphabetically, or in order of registration, because the order will soon be rearranged.

Then, in step 803, the process may determine the number of players to use in the game sessions for the next round. This may be done by consulting the Number of Game Session Entrants parameter described above, but may also involve calculations and adjustments based on the total number of entrants. For example, the process may perform calculations (e.g., dividing the total number of entrants by the various values in the Number of Game Session Entrants parameter) to find an ideal session size. An ideal session size may be a size that allows all sessions to have the same number of players, or for all sessions to differ in size by no more than one player (or any other predetermined value).

When the process has determined the number of players to be in a session has completed, the process may proceed to step 804, and begin filling a new group for the next game session. In step 805, the highest ranking player who has not yet been assigned to a group is assigned to the new group. In step 806, data for the next highest unassigned player (e.g., a potential player) is retrieved, and in step 807, a check is made to see if the potential player has recently played with any of the other players already assigned in the current group. This may be done, for example, by consulting a stored table keeping track of the last time each pair of entrants played together in a game session.

If the potential player has not played too recently with one of the other players in the group, then the potential player is added to the group in step 808. On the other hand, if the potential player has played too recently, then the process skips the potential player and moves on to step 809, where a check is made to see if the current group is full. If the group is not full, the process returns to step 806 to retrieve information for the next unassigned player. If the group is full, the process moves to step 810 to see if there are remaining unassigned players. If there are, the process returns to step 804 to begin filling a new group. One optional step, not shown, may be taken if the last few unassigned players all played too recently with one another. In this situation, the process may simply waive the Window Factor criteria for these players, and group them into one or more final groups. Alternatively, the system may adjust the window factor by a predetermined amount (e.g., reducing it by 1), and begin the process anew.

If all players have been assigned to groups, then the process moves to step 811, where the players will then play a game session. When the game session is completed (e.g., the players finish running a race, complete a football game, finish a deathmatch, etc.), the process moves to step 812 in which players are assigned points based on their performance in the game session. These points may vary depending on game type, with different points being awarded for different game accomplishments. Various types of scoring mechanisms may be used. For example, the tournament server may record game scores for accomplishments made by each player during the most recent game session, cumulative game scores to tally the points accumulated by each player thus far in the tournament, match scores that take into account the number and/or ranking of other players played against (e.g., a point for each player beaten in a given session), and an available cumulative score indicating the maximum number of points the player could have accumulated thus far in the tournament.

The server or console performing the point assignment may also perform calculations for ranking purposes. For example, for the tournament process may calculate a scoring percentage indicating the percentage of the maximum possible score that the player has attained (e.g., by dividing the player’s cumulative game score by the available cumulative score). The process may apply the New Player Factor parameter to adjust a new player’s score before it is used to re-rank the player (in the following step).

Then, in step 813, the process may reorder, or re-rank, the various players based on their total scores and/or their scoring percentage as discussed above. When the players have been reordered, the process may check in step 814 to see if more rounds are to be played. This determination may be based on whether the players have played the requisite number of rounds, or if a player has achieved some other condition for winning, as specified in the tournament parameters. If more rounds are to be played, the process may return to step 804 to regroup the players. If no more rounds need to be played, the process may move to step 815 and end the tournament, as described above with respect to FIG. 6.

FIG. 8c illustrates an example progression of player rankings, groupings and reranking that may occur in the tournament process of FIGS. 8a and 8b. The players may be ranked for round 1 as shown on the left, and grouped into three groups (four players per group), and then reranked as shown on the right based on their performance in round 1. In grouping players for the second round, the four highest reranked players are Players A, E I and B, but since B had just played with Player A in the previous round, the process may skip Player B and add the next highest player, Player F, to Group 1. Players may then be placed in Group 2 for the second round. The example illustrated in FIG. 8c shows the
final player, Player L, being relegated to a new group because that player had played too recently with Player K of the second round’s Group 3. If desired, the process may allow one or more groups at the bottom of the list to play one another regardless of Window Factor restrictions. For example, Player L may simply be matched into Group 3, despite having played recently with Player K. Alternatively, the players at the bottom who cannot be matched may be dropped from the tournament.

0143 As an alternative, one or more groups appearing at the bottom of the rankings may simply be dropped from the tournament, thereby narrowing the field with passing rounds. The dropping may begin after the passage of a predetermined number of rounds (e.g., two rounds), to allow for at least some preliminary ranking so that the dropped groups are more likely to contain the weakest players anyway.

0144 The FIG. 8 method is just one example, and modifications to this type of tournament may be made. For example, the Rank Gap parameter may be used as a second check in steps 807 and 808 to add a player to a group only if the new player is close enough in rank to the other players in the group. The tournament server may also place different priorities on how strict the Rank Gap and Window Factor parameters are to be used, such as requiring the waiver of one or both requirements if the resulting number of required groups would exceed a predetermined threshold, or if there are not enough players to fill groups that meet the parameters’ requirements. The server priorities may waive one priority before waiving another.

0145 As another option, the process may allow for the additional registration of new players after the tournament has started, and the dropping out (e.g., quitting) of players that originally registered. A newly-registered player may simply be added to the pool of available players for the next round, such as before step 803. Alternatively, there may be limitations on when new players are permitted to join (e.g., only after the first predetermined number of rounds, or prohibited in the final predetermined number of rounds, etc.). When adding the new player, the new player may be given a random rank at the outset (as with the initial placement in step 802), or alternatively the player may automatically be ranked in the middle, or at the bottom, of the list. The New Player Factor parameter may be used to adjust a new player’s score to ensure that the new player does not quickly get in over his/her head in the rankings after playing a few rounds. For example, if the scoring percentages are used and a new player wins his/her first round, a New Player Factor might multiply the player’s scoring percentage by 0.25 when the player has only played one round, so that a new player will not ascend too quickly after winning one round. Similarly, the New Player Factor may vary depending on how many rounds the player has played (e.g., using 0.5 after two rounds, and 0.75 after three, and 1.0 after four rounds), so that the player’s full score is credited after a predetermined number of rounds.

0146 The FIG. 8 tournament process may be used to accommodate large numbers of players, and can be used to run a continuous tournament (e.g., one in which there is no defined number of rounds). FIG. 9 illustrates another tournament process that may be used in addition to that of FIG. 8. In the FIG. 9 process, a leaderboard qualification period may be used to initially qualify and seed the tournament entrants. The process begins at step 901, where a tournament creator or administrator (which may be a game developer, player, etc., as discussed above) initially defines the parameters of the tournament. The parameters may be similar to those discussed above, but other parameters may also be used. For example, the tournament may use an elimination bracket structure in which winners advance and losers drop out, and such tournaments may include a separate parameter identifying the number of participants to appear in the final, championship, game, and the number of winners that advance from earlier-round games. The tournament parameters may also identify the number of tournament levels that will be used. For example, a tournament may be defined as having three levels (e.g., “Gold”, “Silver” and “Bronze”) to allow the highest, second highest, and third highest ranking group of players compete with one another. Each of these levels may be operated as individual instances of the tournament, allowing players of different caliber to compete in their own tournaments. The separate instances may be initiated, for example, after qualification is completed and the entrants are divided into ranked groups based on their qualification.

0147 The tournament may also include scheduling information for the qualification round(s), such as the opening time/date and duration. The tournament may also include a Challenge Definition parameter that sets forth the type of challenge that will be used to qualify participants. A Challenge Definition data structure may generally indicate a type of feat that a player must accomplish in order to qualify for the tournament, and may vary depending on the game. The Challenge Definition may specify one or more game settings for the qualifying challenge, such as a location setting (e.g., race track for a racing game, a battlefield map for a war game, a game level or stage, etc.), difficulty setting (e.g., easy, medium, hard, etc.), game condition (e.g., type of car for a race, available weapons on a map, weather setting), etc. The Challenge Definition is not limited to just one event. Instead, the definition may specify a number of qualifying events in an ordered list, such as a sequence of different races or stages, with different conditions.

0148 For some games, the Challenge Definition may include a data file that can be downloaded to a game console 402 to automatically configure a game title program on the console to participate in the challenge. For example, if a race game’s Challenge Definition parameter requires that a 4-lap race be run on the Laguna Seca racetrack on a sunny day, using rear-wheel drive cars having no more than 200 horsepower, and against 5 computer-controlled (AI) opponents set on a race game’s “medium” difficulty setting, the tournament server may include a tournament challenge download file that will, when downloaded to the player’s console, automatically configure the game running on the console to run the type of race specified in the download file. The download file may also be used by the game running on the console 402, to restrict the player’s choice of cars to those permitted by the Challenge Definition.

0149 When the tournament parameters are defined, the tournament’s server may then begin to execute a process that will handle the tournament. The process may allocate (or otherwise reserve) storage for a leaderboard data structure that will be used to hold players’ qualification results, and the leaderboard’s ID may be one of the parameters defined...
in step 901. The leaderboard ID may include a name or handle for the data structure, and a memory address location indicating where it is stored and how it can be retrieved. The leaderboard IDs may be used for scoring arbitration.

[0150] When the tournament parameters have been set in step 901, a computer process may then announce the tournament to players in step 902. The announcement may occur via electronic communication, such as e-mail, SMS, etc., and may include sending electronic messages to game consoles 402. Announcements may also be made using other forms of communication, such as television advertising, radio, telephone, magazines, etc. The announcement informs players of the parameters of the tournament, the type of challenge that must be accomplished to qualify for the tournament (e.g., the Challenge Definition), and the time period for the qualification period (e.g., start time and end time).

[0151] After the announcement in step 901, and at whatever scheduled time is set in the tournament parameters (or immediately, if no time is scheduled), the process opens the qualification period at step 903. During this period, players may download a Challenge Definition configuration file, if available, and play the requisite challenge (e.g., run the specified race). The game console 402 may record a player’s progress and accomplishments (e.g., race time), and may report a score through secure transmission back to the tournament’s server, such as Tournament Server 434, for inclusion on the tournament’s leaderboard. In addition to the secure transmission checking discussed above, this reported score may also undergo a verification process to help further avoid having fraudulent scores posted. The verification process may include a comparison of a reported score with a predetermined score limit or maximum expected score (which may be additional tournament parameters defined above), or with results reported by a different game console.

[0152] After receiving a player’s score, or in response to a player’s request, the server may query the leaderboard to determine a tournament ranking based on the player’s result and other players’ qualification scores. The server may send a message to the player informing the player of how well he/she performed (e.g., “You have logged the 75th fastest time for this challenge”). The message may also inform the player as to what the player has qualified for (e.g., “Your score qualifies you for the gold level tournament,” or “Your score qualifies you for the 10th seed”). If the player desires, the player may try the qualification event again and post another score, hoping to improve his/her position. To avoid duplicative scores, the server may obtain an identity (e.g., login identity) of the player posting a score (e.g., transmitted with the score), and check to see if the player has previously posted a qualification score. If the player has, the server process may automatically replace the prior score with the newer one, or it may do so only if the newer score would result in a higher rank than the old one, or if the user requested to replace an old score with a newer one when he/she submitted the score. Alternatively, players need not be given unlimited opportunity to post qualifying scores. For example, the tournament may restrict each user to just one qualification entry, or a limited predetermined number of qualification entries. The player may be given an opportunity to have a game session score ignored (e.g., treating it as a practice round) instead of being posted. Additionally, the system may display an option to users to purchase additional qualification attempts, and users may have a predetermined account (e.g., a credit card account, or a subscription account) debited for the cost of having another try at posting a higher qualification score.

[0153] As more and more players qualify during the qualification period, many players may move down in the rankings. The tournament server process may automatically notify such players when their ranking is changed, or if their ranking changes by a predetermined amount (e.g., a number of places, a percentage of participants, etc.), or if their ranking drops them out of qualification for a particular tournament instance (e.g., if their prior qualification used to qualify for the “Gold” tournament, but now only qualifies for the “Silver” tournament). The notification may be in any communication, similar to the announcement, and may be an electronic message sent to the player’s console 402. The message may appear as an in-game communication on the player’s console, displayed using a game program’s user interface. Other notifications, such as e-mail to a computer, may also be used. The actual type of notification, and conditions on which they are sent, may be specified as additional tournament parameters.

[0154] When qualification is completed, the tournament server may then consult the leaderboard data structure in step 904, and determine how many players have actually qualified for the tournament by comparing the posted results with the requirements for qualification. The requirements for qualification may be specified in, or derived from, the tournament parameters. An example of such a derivation may use a calculation based on entered tournament parameters, such as the desired number of players per game (P), the number of players to be in the final game (P_R), the number of winners who should advance from each game (W) (e.g., 2, 3, 4, etc.; players advance from each game), and the number of rounds desired (R), to calculate the total number of players (P_T) that can participate in one instance of a tournament:

\[ P_T = P_R \left( \frac{W}{P} \right)^{R-1} \]

[0155] To determine which players qualify, the tournament process may perform the above calculation, and then consult the leaderboard to accept the top P_T number of players as qualified for the tournament. Of course, this calculation may have been performed earlier, such as in the tournament definition step 901 and/or qualification period 903. If multiple tournament instances are supported (e.g., as indicated by the tournament parameters), then the tournament process may accept the next P_T entries in the leaderboard as qualifying for the next tournament, and so on until the total number of tournament instances has been filled, or until there are no more unassigned entries in the leaderboard. In this manner, the tournament process divides the qualifiers (or entrants) into the various tournament levels.

[0156] The example calculation above is not limited to determining a total number of players per tournament. Instead, any one of the variables may be calculated based on other provided variables. So, for example, if a tournament administrator decided to leave the number of rounds (R) open, the equation may be rearranged to calculate that value.
based on the other values (e.g., conduct as many rounds as it takes to have a tournament for all players and meeting the specified player-per-game and winner-per-game values). Additionally, a tournament fill ratio may be defined in the parameters (e.g., 20%) to require that a certain number or percentage of posted scores will qualify for a tournament. When the qualification period ends, and the tournament process can consult the leaderboard to determine the overall number of entries received, the process can determine how many (e.g., 20%) of the entries will be in the tournament (e.g., P = 20 if ratio is 20% and 100 entries received), and can then adjust the other values (e.g., R, P, etc.) to accommodate the fill ratio.

[0157] In step 905, the tournament process may then schedule the various tournament(s) and tournament round(s), based on the calculated values (if any) and any schedule parameters set in the tournament parameters. The process may also perform seeding in step 906 to fill tournament brackets in elimination tournaments. This seeding may be done dynamically using the grouping approach described above to keep players of close skill playing one another each round (e.g., the top X players play in one game, the next X players play in a second game, etc., with seeding occurring after each round and as late as possible), or the seeding may be done to try and ensure that the higher-seeded players do not face other high-seed players until the later or final rounds. This may be done generally by starting with the championship game, determining the number of semifinal matches needed to fill the final match, and recursively performing the same analysis for the semifinal matches out until all rounds have been defined. When the rounds of matches have been defined, the seeding process may then step from opening match to opening match, placing the top seed players in numbers sufficient to fill the expected winners, and then filling out the rest of the matches with the lower seed players. This kind of seeding may be statically done, in which case players carry their seed with them throughout the tournament, and are not reseeded in between rounds. As an additional alternative, this seeding may be conducted anew after each round of play, where only those players that are present and ready to play (e.g., have checked in with the server handling the tournament instance). So, for example, if a number of players quit or drop out after the first round, the tournament may be re-seeded for the second round to minimize idle players (e.g., players whose opponents, according to the original seeding, are no longer playing).

[0158] FIG. 10a illustrates an example of a statically-seeded bracket that may be generated when the tournament server follows the process shown in FIG. 10a. Although various values are shown in the intermediate rounds of the FIG. 10a bracket, the end result of the process is to identify the opening round bracket matchups, the overall number of rounds, and the arrangement of the matches (e.g., which earlier matches supply a later match with its competitors). As shown in FIG. 10b, the process begins in step 1001 by considering the final round. The process may consult the tournament parameters to determine how many competitors are to be in the final round, and may fill the final round in step 1002, the final round may be filled with the top seeds. In the FIG. 10a example, the final round has been defined to have eight competitors, so the top eight seeded players are presumed to appear in the final round. Again, this is just a presumption for purposes of seeding the opening rounds, as the top seeded players are not required to actually survive to the final round.

[0159] Then, in step 1003, the system may check to see how many prior feeder matches are needed in the prior round (e.g., the semifinal round) to fill the current round (e.g., the final round). For example, if the final round needs eight players, and the tournament parameters indicate that four players advance from each match, then the earlier round (the semifinal round) would need two matches to provide those eight players. In step 1004, the players from the current round are distributed across the various matches needed in the earlier round. In step 1005, the remainder of the matches in the earlier round are filled with the next highest available seeded players.

[0160] In step 1006, a check may be made to determine whether all entrants in the tournament have been placed. If they have not, then another earlier round will be needed in the tournament. The process may, in step 1007, then shift the focus to the next earliest round, and return to step 1003 to perform the same steps to fill the matches of the earlier round. So for example, the system may set the “current round” in step 1007 to be the semifinal round, and may return to step 1003 and begin to fill the brackets for the next earlier round (e.g., the quarterfinal round). The process may continue until all entrants have been placed, and the last round addressed in the process provides the opening round seedings.

[0161] FIG. 11 illustrates another example of a tournament in which six players appear in the final round, nine players play in the prior rounds’ matches, and three players advance from each prior match. The earliest rounds indicate the end result of the seeding process.

[0162] The various tournament instances may then be instantiated, and the tournament(s) run in step 907 to determine their outcomes. The actual running of the tournaments may occur as described above, with separate computer program processes on the same or different servers administering the various tournament instances, game sessions being run on consoles and reported to the server, score arbitration, etc.

[0163] FIG. 12 illustrates an example user interface map that may be used, for example, in a game to access tournaments of the type shown in FIGS. 8a-8b and 9. The main menu 1201 of the console may offer a number of tournament options, including a tournaments page option. The tournaments page 1202, displayed in response to selecting that option, may include a menu of tournament options. For example, the tournaments menu 1202 may display a list of tournaments for which the player has already registered, the player’s qualification rank, the number of entries, the tournament’s current status (e.g., registration, qualifying, round X, etc.). The tournaments menu 1202 may also include selectable options for entering one of the listed tournaments, finding a tournament and for creating a tournament. Upon choosing one of the listed tournaments, the user may be presented with a tournament lobby screen 1203 for the selected tournament.

[0164] The tournament lobby screen 1203 may display additional information for the selected tournament, such as current status, the player’s next opponent, status of oppo-
nents, etc. The tournament lobby screen 1203 may include additional menu options for playing a game in the tournament, viewing a game history for prior matches in the tournament, etc. Upon selecting a corresponding menu option, the game console may display a play game screen 1204 to begin playing a round or match in the tournament, or a game history screen 1205 to display information regarding a player's prior performance in the tournament. This information may include statistics for players, opponents, teams, etc. The history screen 1205 may also display other player online performance across multiple tournaments and game types, such as by querying data from a leaderboard on Statistic Server 426.

[0165] The tournament lobby screen 1203 may also include an entrant list option, selection of which may display entrant list 1206. The entrant list screen 1206 may display identification and statistical information regarding entrants in the tournament, entrants who are scheduled to play in the next round, entrants who played in the previous round, etc.

[0166] After playing a match or round, the system may display a report results screen 1207 to display the outcome of the match. This may include statistics regarding players' performance in the match that was just played.

[0167] From the tournaments menu 1202, players may also be given the option of searching for tournaments. Upon selecting this option, the player may view a find tournaments screen, which may include prompts for the user to enter, or select, search criteria that may be used to locate tournaments on the online system, such as those running on one or more Tournament Servers 434, or those using the leaderboards. The criteria may include game title, tournament name, game type, tournament type, tournament host, or any other desired search criteria. Upon running the search, the system may display a list of tournaments for this user, indicating qualification details (e.g., whether the user qualified, the entrant qualifier scores, etc.) and a schedule of when the tournaments are set to occur.

[0168] The tournament menu 1202 may also include an option to allow players to create new tournaments. Upon selection, the user may be shown a create tournament screen 1209, which may prompt the user for criteria to define a new tournament. The criteria may include any of the features described above, such as the tournament parameters, game title, etc. One or more additional game settings screens 1210 may also be used for this purpose.

[0169] The tournament menu 1202 may also include a leaderboard qualified tournament option, and the selection of this option may display a leaderboard screen 1211. The leaderboard screen 1211 may list leaderboard tournaments, and contain additional data regarding leaderboard qualified tournaments, such as the name, ID, current state, current challenge, challenge/qualification period, etc. The leaderboard screen 1211 may include an option to display additional explanation and detail on the qualification for a particular displayed tournament. The qualification details screen 1212 may include additional description to explain how the player may qualify (e.g., "Race the Laguna Sea race track using a 4WD car with less than 301 Hp, and post one of the top 150 times to qualify for the Gold level tournament"). The user may choose to make a qualifying attempt, resulting in the game screen 1213, and after the attempt the user may view a report results screen 1214 displaying information regarding the player's qualification attempt. The results screen 1214 may also display information retrieved from a leaderboard showing how the player's resulting score ranks among others, and information identifying whether the player qualified for the tournament.

[0170] Upon selecting a listed tournament in the leaderboard screen 1211, the console may display an elimination details screen 1215 for the tournament. The elimination details screen 1215 may list the tournament status, the player's standing/ranking, and if the player is still in the tournament, the next scheduled game for the tournament. When the player is ready to begin the next round, the player may select a corresponding option from the details screen 1215 and enter the tournament lobby 1216, where the player may await the entry of his/her opponents for the next round. From the lobby 1216, the player may choose to view the bracket 1217 for the tournament, the game settings and parameters 1218, or statistics 1219 regarding the player, his/her teammates, friends, other competitors in the tournament, and upcoming opponents.

[0171] When the player's opponent has checked in, and when both players are ready, the player may view the game screen 1220 to play the match, and a post-game results screen 1221 showing the results of the match (as described above).

[0172] The various features described above are mere example implementations of various concepts, and modifications are possible as desired. For example, the references to a player above may refer to a single person, or it may refer to a team of people playing their games together. Some tournaments may support teams, or clans, by allowing the various players to participate together (e.g., in a single race) as a team. Scores for team members may be aggregated for ranking comparison.

[0173] Another possible modification deals with dropouts and no-shows for tournament participants. If a player enters a tournament by registering and/or qualifying, but fails to complete the tournament (e.g., dropping out after starting, or never showing up at the scheduled time, etc.), that player can be treated as if the player had accumulated a zero score, or a loss.

[0174] As another possible modification, the various screens shown in Fig. 12 and features described herein may be used for local, non-networked play as well. For example, a multiplayer tournament may be hosted by a single console, and the various features implemented by the console.

[0175] The features described above are preferably encoded in computer software as executable instructions that can be executed on a computing device, such as a personal computer or video game console, to result in the display of the screens shown in the figures. The executable instructions may be stored on a computer-readable medium, such as one or more computer disks, RAMs, CD-ROMs, DVDs, game cartridges, etc. Also, although various features are described above, it is not necessary to practice them all in the same embodiment. Instead, various combinations and subcombinations may be implemented as desired, and the true scope of the present invention should only be limited by the claims that follow.

[0176] Although the subject matter has been described in language specific to structural features and/or methodologi-
cal acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

We claim the following:
1. A tournament method, comprising the steps of:
   storing, in one or more computer-readable storage devices, a set of qualification requirements for a multi-round tournament, wherein rounds of said tournament comprise one or more matches involving a plurality of competing players, wherein a subset of players is to be eliminated by each match, and a subset of players advances from each match to compete in a later round of said tournament;
   electronically receiving a plurality of player qualification scores;
   ranking a plurality of players according to said plurality of qualification scores and said qualification requirements;
   determining a variable tournament parameter for said tournament based on said ranked plurality of qualification scores, wherein said variable tournament parameter varies between instances of said tournament, and is one of the following parameters:
   i) a number of players in a final round of said tournament;
   ii) a number of players advancing from a non-final round of said tournament to a subsequent round in said tournament;
   iii) a number of players in each match of said tournament; and
   iv) a number of rounds in said tournament; and
   conducting a first instance of said tournament on one or more computing devices using said determined variable tournament parameter after said variable tournament parameter is determined.
2. The method of claim 1, wherein said variable tournament parameter is said number of players in a final round of said tournament.
3. The method of claim 1, wherein said variable tournament parameter is said number of players advancing from a non-final round of said tournament.
4. The method of claim 1, wherein said variable tournament parameter is said number of rounds in said tournament.
5. The method of claim 1, further comprising a step of conducting a second instance of said tournament, wherein a first portion of said ranked players is assigned to said first tournament instance, and a second portion of said ranked players is assigned to said second tournament instance.
6. The method of claim 5, wherein said second tournament instance is conducted while said first instance is conducted.
7. The method of claim 5, wherein said first portion of said ranked players is ranked higher than said second portion of said ranked players.
8. The method of claim 1, wherein said step of determining includes a step of applying a tournament fill ratio to said ranked plurality of qualification scores, said tournament fill ratio identifying a proportion of said ranked players who will participate in said tournament.
9. A tournament method, comprising the steps of:
   receiving a plurality of scores from a plurality of players for performance during a plurality of matches of a tournament; and
   ranking said players according to a ratio between each player’s cumulative match score and said player’s accumulated cumulative match score, wherein said cumulative match score is a total of points earned by said player during said player’s matches, and said available cumulative match score is a total of the maximum possible points said player could have earned in said player’s matches.
10. The method of claim 9, further comprising the following steps:
   adding a new player to said tournament after a first round of said tournament has completed;
   including said new player’s score in said steps of receiving and ranking, wherein said step of ranking further includes a step of applying a new player factor to said new player’s score, said new player factor being based on a number of tournament matches said player has played.
11. The method of claim 9, wherein said step of ranking is performed after each of a plurality of rounds in said tournament, and further comprising the step of grouping said players for matches in a second round of said tournament based on a ranking of said players after a first round of said tournament.
12. The method of claim 11, wherein said step of ranking omits players who have quit the tournament before the tournament’s conclusion.
13. An online multiplayer tournament method for an interactive video game, comprising the steps of:
   conducting a session of an interactive video game on a computing device for a user;
   transmitting results of said session to an online multiplayer multi-round tournament server over a communication network;
   receiving a request by said user to display a listing of online multiplayer multi-round tournaments, for said interactive video game, for which said user has qualified; and
   responsive to said request, displaying a list of online multiplayer multi-round tournaments, for said interactive video game, for which said user has qualified; said list being displayed on a display device associated with said computing device.
14. The method of claim 13, wherein said displayed list identifies said user’s qualification rank for one or more tournaments.
15. The method of claim 13, wherein said displayed list identifies a current status of one or more tournaments.
16. The method of claim 13, further comprising steps of:
   receiving one or more search criteria; and
   conducting a search for tournaments satisfying said one or more search criteria; and
displaying a list of tournaments satisfying said one or more search criteria.

17. The method of claim 16, wherein said step of conducting is performed responsive to a user selection from said displayed list of tournaments satisfying said one or more search criteria.

18. The method of claim 13, wherein said displayed list further displays schedule information for one or more tournaments in said list.

19. The method of claim 13, further comprising a step of displaying an option for said user to create an online multiplayer multi-round tournament.

20. The method of claim 13, further comprising a step of displaying statistical information for tournament entrants for one of said tournaments from said displayed list.

* * * * *