US 20120021835A1

(19) United States

(12) Patent Application Publication

Keller et al.

(10) Pub. No.: US 2012/0021835 A1

(43) Pub. Date: Jan. 26, 2012

(54) SYSTEMS AND METHODS FOR SERVER BASED VIDEO GAMING

(75) Inventors: **John Keller**, Burleson, TX (US); **Jonathan Strietzel**, Long Beach, CA (US)

(73) Assignee: **IPRD LABS LLC**, Long Beach, CA (US)

(21) Appl. No.: **13/006,387**

(22) Filed: **Jan. 13, 2011**

Related U.S. Application Data

(63) Continuation of application No. 11/057,394, filed on Feb. 11, 2005.

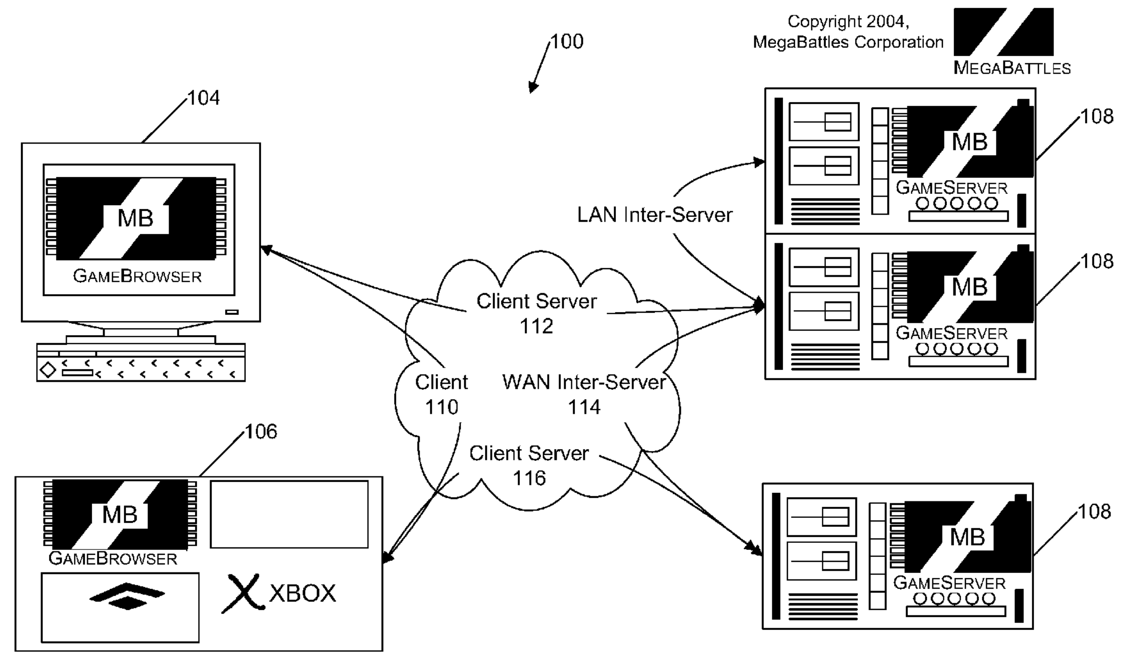
Publication Classification

(51) **Int. Cl.**
A63F 9/24 (2006.01)

(52) **U.S. Cl.** **463/42**

(57) **ABSTRACT**

A server based videogame system comprises a game authority configured to store the resources and scripts associated with a plurality of video games. Users can access the games, which are hosted by the game authority using various client devices. Only portions of a game are then served to the user's client device when the user is playing a game. The user's client device is configured to predict what resources will be need soon based on a game page provided by the game authority that lists all of the resources and scripts associated with the game. The client device can then request resources and scripts before they are needed in order to maintain game flow.



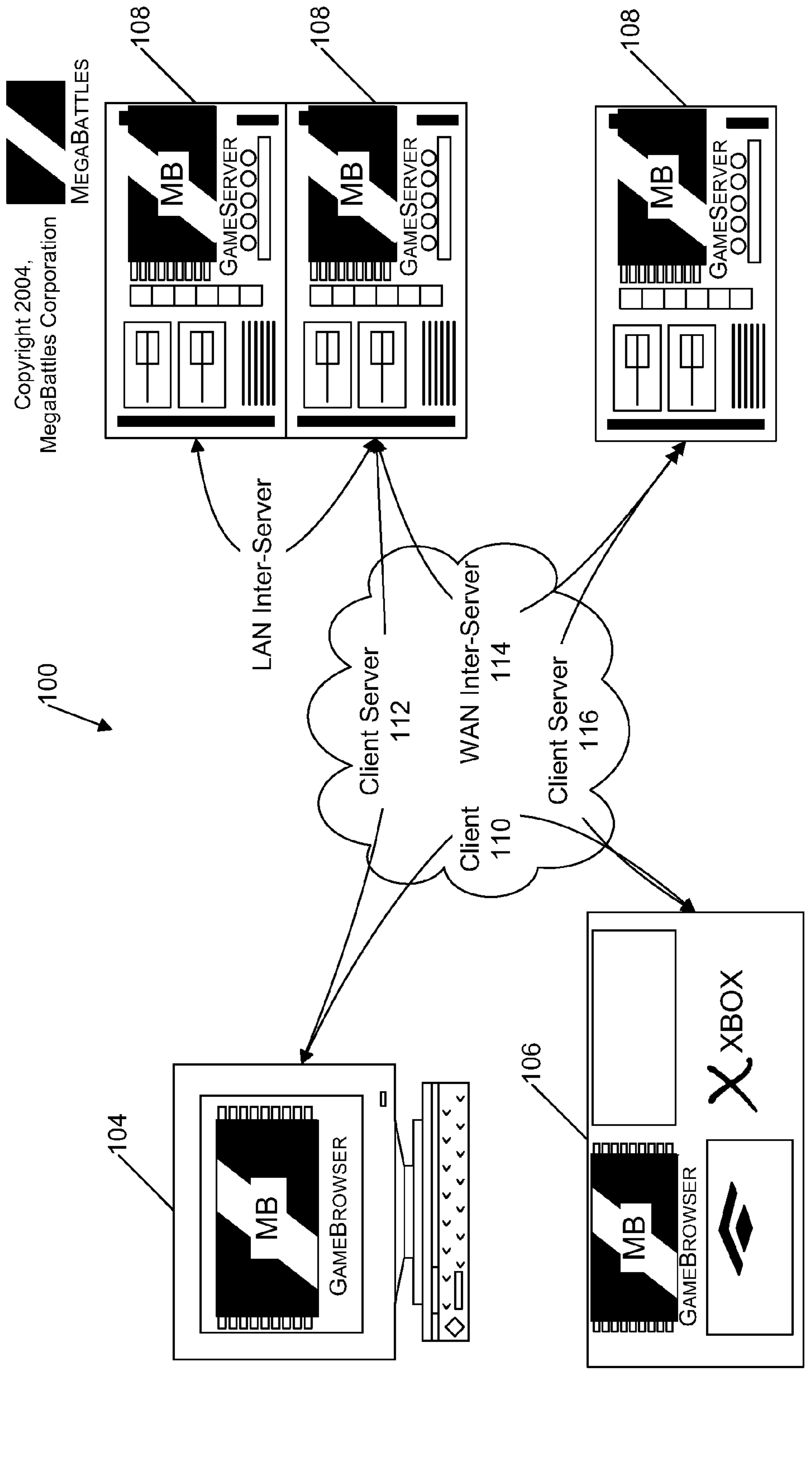

Copyright 2004,
MegaBattles Corporation
MegaBattles
108
108
108
MB
GameServer
MB
GameServer
MB
GameServer
LAN Inter-Server
100
Client Server
112
WAN Inter-Server
114
Client Server
116
Client
110
104
MB
GameBrowser
106
MB
GameBrowser
XBOX

FIG. 1

FIG. 2

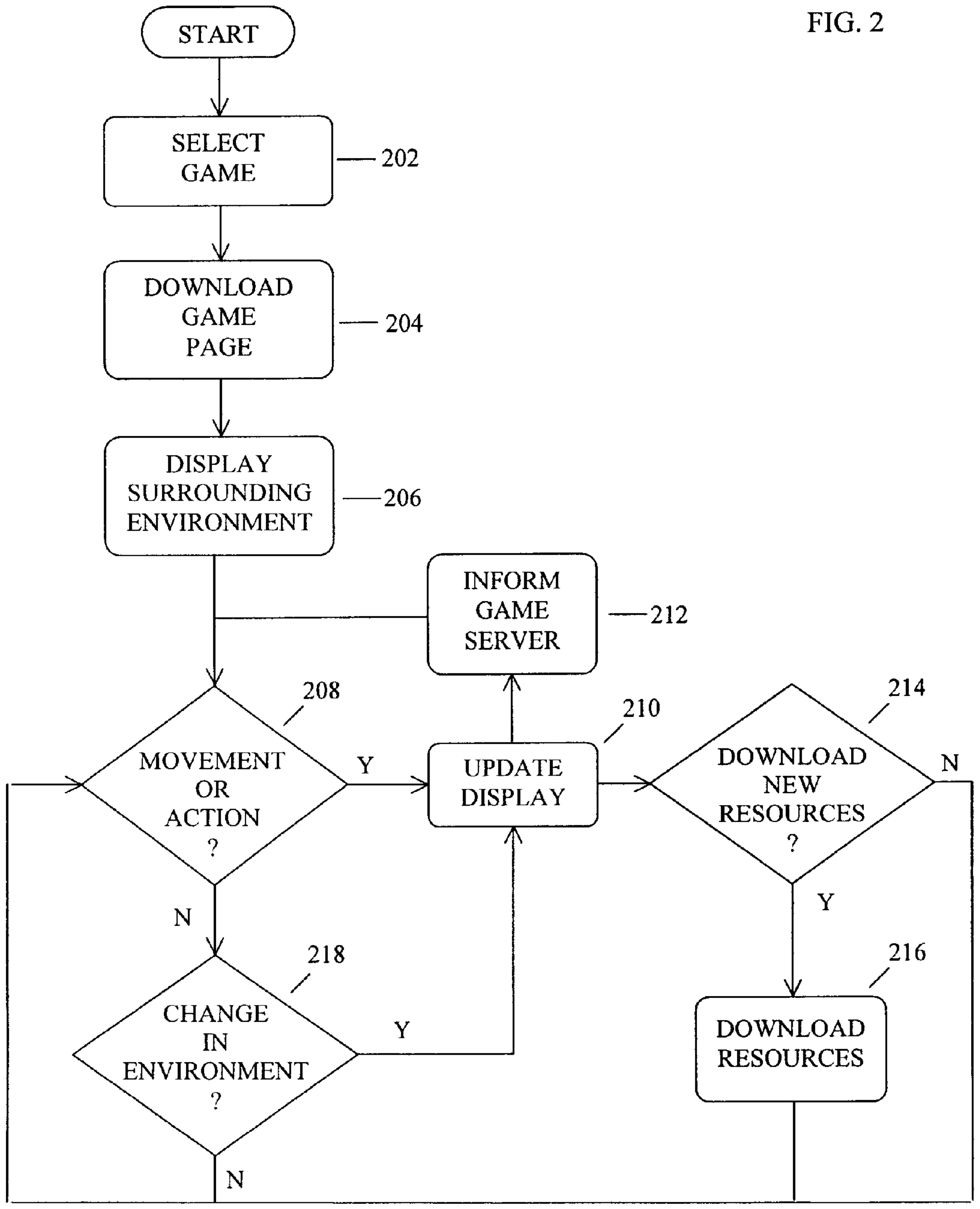
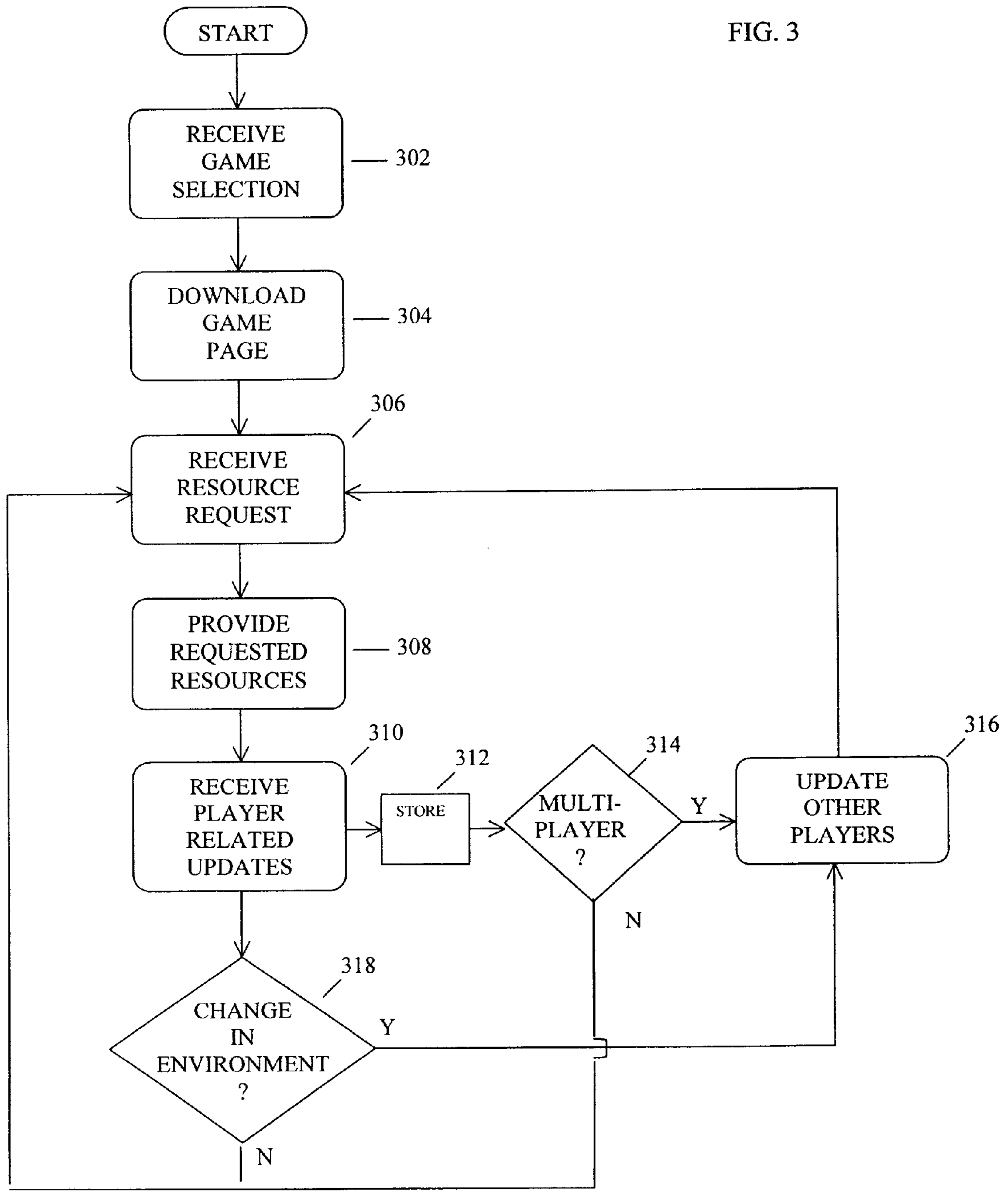

START
SELECT GAME
202
DOWNLOAD GAME PAGE
204
DISPLAY SURROUNDING ENVIRONMENT
206
INFORM GAME SERVER
212
MOVEMENT OR ACTION ?
208
Y
N
UPDATE DISPLAY
210
DOWNLOAD NEW RESOURCES ?
214
N
Y
CHANGE IN ENVIRONMENT ?
218
Y
N
DOWNLOAD RESOURCES
216

FIG. 3


START
RECEIVE GAME SELECTION
302
DOWNLOAD GAME PAGE
304
306
RECEIVE RESOURCE REQUEST
PROVIDE REQUESTED RESOURCES
308
310
RECEIVE PLAYER RELATED UPDATES
312
STORE
314
MULTI-PLAYER ?
Y
N
316
UPDATE OTHER PLAYERS
318
CHANGE IN ENVIRONMENT ?
Y
N

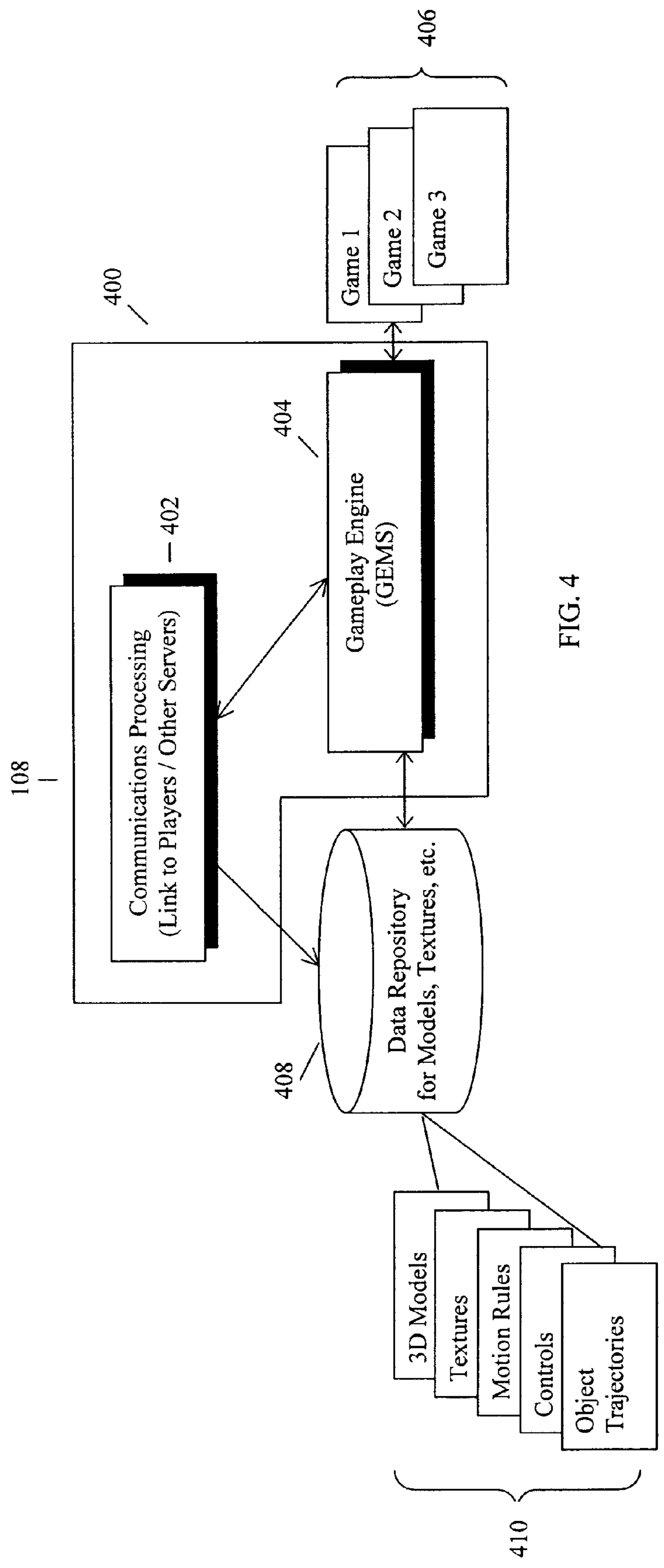

108
400
Communications Processing (Link to Players / Other Servers)
402
404
Gameplay Engine (GEMS)
406
Game 1
Game 2
Game 3
408
Data Repository for Models, Textures, etc.
3D Models
Textures
Motion Rules
Controls
Object Trajectories
410

FIG. 4

FIG. 5

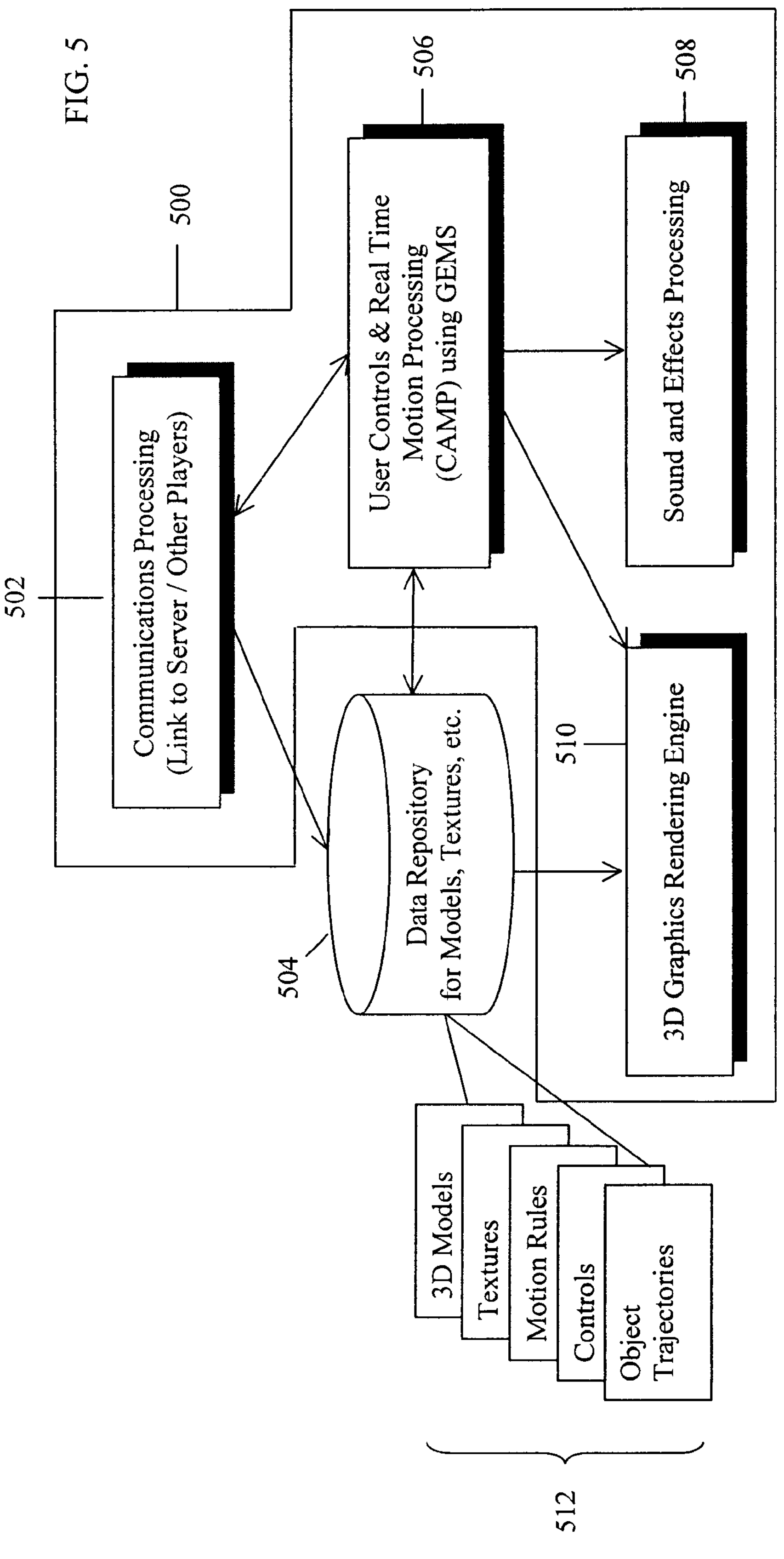
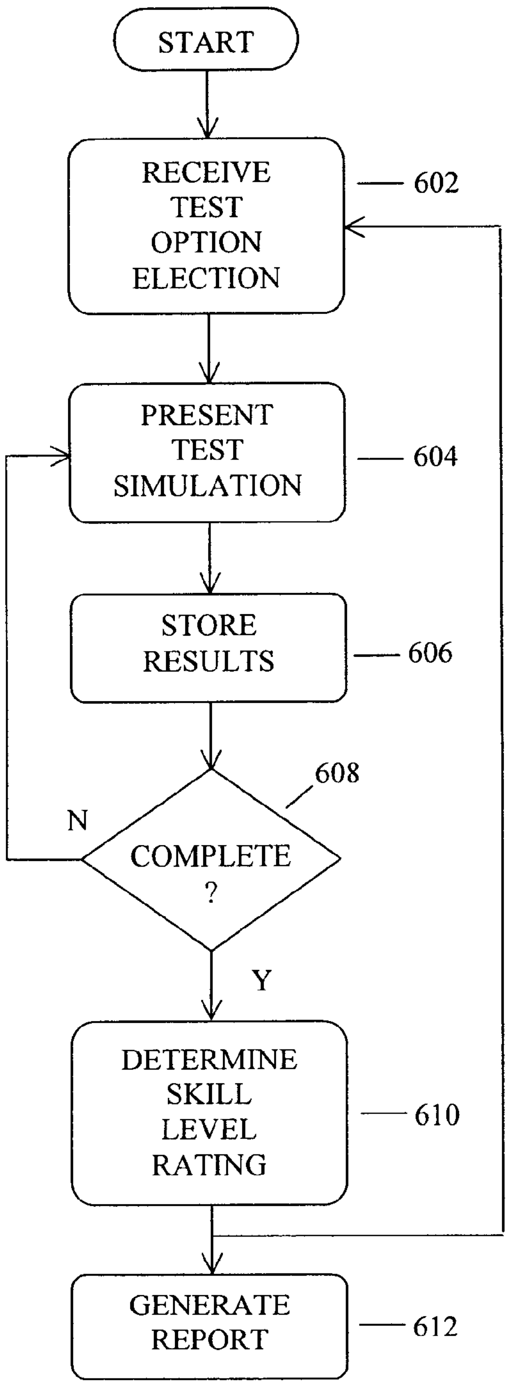

500
502
Communications Processing (Link to Server / Other Players)
506
User Controls & Real Time Motion Processing (CAMP) using GEMS
508
Sound and Effects Processing
504
Data Repository for Models, Textures, etc.
510
3D Graphics Rendering Engine
3D Models
Textures
Motion Rules
Controls
Object Trajectories
512

FIG. 6


START
RECEIVE TEST OPTION ELECTION
602
PRESENT TEST SIMULATION
604
STORE RESULTS
606
608
N
COMPLETE ?
Y
DETERMINE SKILL LEVEL RATING
610
GENERATE REPORT
612

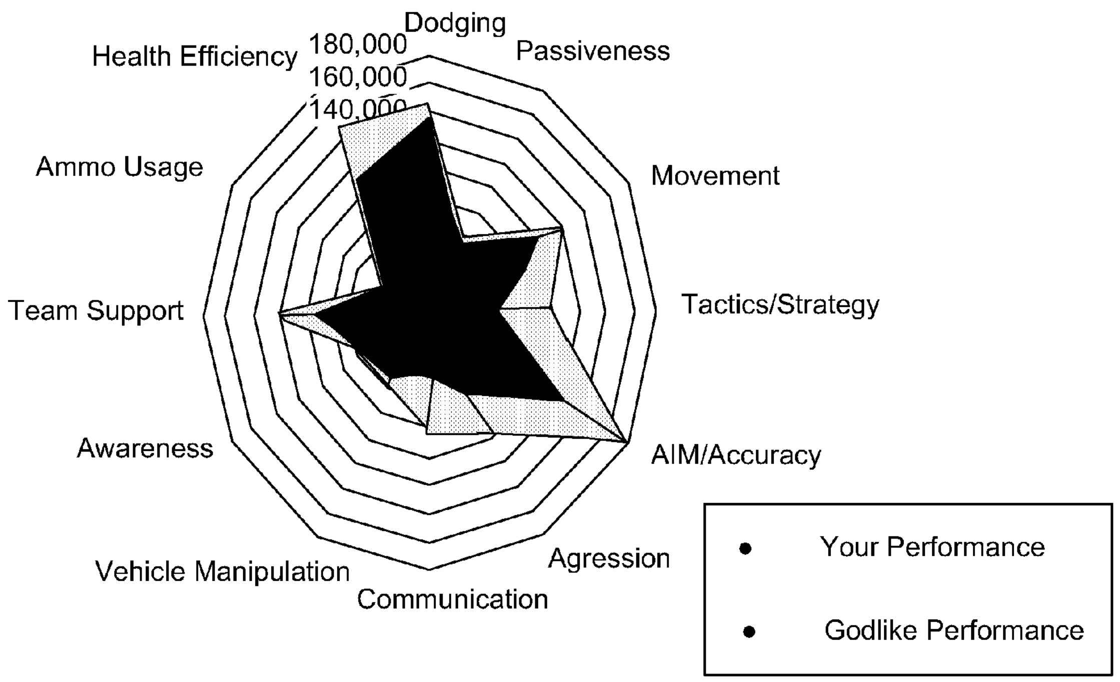

Dodging
Passiveness
Movement
Tactics/Strategy
AIM/Accuracy
Agression
Communication
Vehicle Manipulation
Awareness
Team Support
Ammo Usage
Health Efficiency
180,000
160,000
140,000
● Your Performance
● Godlike Performance

FIG. 7

SYSTEMS AND METHODS FOR SERVER BASED VIDEO GAMING

RELATED APPLICATIONS INFORMATION

[0001] This application claims priority as a continuation under 35 U.S.C. 120 to U.S. patent application Ser. No. 11/057,394, filed Feb. 11, 2005, and entitled "Systems and Methods for Server Based Video Gaming," which is incorporated herein by reference as if set forth in full.

[0002] This application is also related to U.S. patent application Ser. No. 11/058,093, filed Feb. 15, 2005, and entitled "Systems and Methods for Developing Video Games in a Server Based Video Gaming Environment," and U.S. patent application Ser. No. 11/058,094, filed Feb. 15, 2005, and entitled "Systems and Methods for Providing Virtual Camera Views in a Server Based Video Gaming Environment," and U.S. patent application Ser. No. 11/058,095, filed Feb. 15, 2005, and entitled "Systems and Methods for Providing Player Skill Rankings in a Server Based Video Gaming Environment," all of which are incorporated herein by reference as if set forth in full.

BACKGROUND

[0003] 1. Field of the Invention

[0004] The inventions described herein relate generally to gaming videogames and more particularly to systems and methods for a server based videogame system.

[0005] 2. Background of the Invention

[0006] Computer video gaming has grown more and more popular with an ever broader segment of the population. Modem videogames can be played on a variety of client devices. For example, many games can be loaded on a home, or personal computer, while others are designed to be played on a customized game console that typically interfaces with the users television. The customized game consoles tend to have proprietary technology that requires that games be designed specifically for the particular game console. Even games designed for a personal computer typically require that the game be designed for specific operating systems.

[0007] Designing games for different platforms is not necessarily a trivial endeavor. Thus, the game developer must often commit to a specific platform and dedicate development resources to developing a game for the selected platform. In the case of custom game consoles, the game developer is often required to enter into a deal with the console manufacturer before the game can be sold to the public. Typically, the console manufacturer has tremendous leverage in negotiating deals with game developers, because they control access to the console required for the user to play the game.

[0008] The Internet has enabled networked versions of many games, where users can interact within the game environment via network connections. This interaction can comprise users collaborating to play the game, or users playing against each other.' To allow such collaboration, each user must purchase, or posses a copy of the game. Thus, the users will typically be using the same type of game console. The users then log onto, or otherwise access, a central game server. Each user then begins their version of the game. Each user's game console then communicates with the game server providing information related to the user's movements, actions, results, effects, etc., within the game. The server then communicates this information to the game console associated with the other users participating in the game. The game environment as seen, or experienced by these others is then updated accordingly by the user's game console.

[0009] The game server can also provide, or enable, communication between the user's game consoles. For example, Voice Over Internet Protocol (VoIP) communication links can be established between the user's game consoles to allow the users to communicate with each other within the game environment.

[0010] As with many digital media based industries, pirating of the games is a problem within the videogame industry.

SUMMARY OF THE INVENTION

[0011] A server based videogame system comprises one or more game authorities configured to store a plurality of videogames that can be accessed by users using client devices at locations remote from the game authority.

[0012] In one aspect, the game authority comprises a game server configured to server portions of the game scripts and resources to the client devices as needed.

[0013] In another aspect, the client devices include game browsers that act similarly to web browsers by requesting resources from the game authority as needed using URL requests.

[0014] These and other features, aspects, and embodiments of the invention are described below in the section entitled "Detailed Description."

BRIEF DESCRIPTION OF THE FIGURES

[0015] Features, aspects, and embodiments of the inventions are described in conjunction with the attached drawings, in which:

[0016] FIG. **1** is a diagram illustrating a server based videogame system in accordance with one embodiment;

[0017] FIG. **2** is a flow chart illustrating an example method for implementing a server based video system from the perspective of a client device included in the system of FIG. **1**;

[0018] FIG. **3** is a flow chart illustrating an example method for implementing a server based video system from the perspective of a game authority included in the system of FIG. **1**;

[0019] FIG. **4** is a diagram illustrating an example embodiment of a game authority that can be included in the system of FIG. **1** and configured to implement the method of FIG. **3**;

[0020] FIG. **5** is a diagram illustrating an example embodiment of a client device that can be included in the system of FIG. **1** and configured to implement the method of FIG. **2**;

[0021] FIG. **6** is a flow chart illustrating an example method for providing skill level rankings to game players within the system of FIG. **1** in accordance with one embodiment; and

[0022] FIG. **7** is a graph illustrating an exemplary player skill ranking that can be produced using the method of FIG. **6**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

1. Overview

[0023] FIG. **1** is a diagram illustrating a server based video gaming environment **100** in accordance with one embodiment of the systems and methods described herein. System **100** comprises one or more game authorities **108** interfaced with a plurality of client devices, of which client devices **104** and **106** are illustrated by way of example, via network **102**. Server authorities **108** host a plurality of videogames that can be accessed by users using client devices **104** and/or **106**.

[0024] The term "authority" is intended to refer to all of the systems, both hardware and software, required to perform the functionality described herein. Thus, a game authority **108** can comprise one or more servers, computers, routers, databases, user interfaces, software programs and routines, etc., as required. Moreover, as illustrated several game authorities **108** can be included in system **100** depending on the embodiment. Multi-authority embodiments are described in more detail below. Each game authority **108** can comprise a game server configured to implement many of the functionalities described herein. The game server is described in more detail below as well.

[0025] Network **102** can comprise one or more Metropolitan Area Networks (MANs), Wide Area Networks (WANs), Local Area Networks (LANs), and Personal Area Networks (PANs). Moreover, the network, or networks, comprising network **100** can be wired or wireless or some combination thereof. Network **100** can also comprise the Internet. Network **102** can enable communication between game authorities **108** (link **114**), between game authorities **108** and client devices **104** and **106** (links **112** and **116**), and between client devices, e.g., between devices **104** and **106** (link **110**).

[0026] In certain embodiments, communications flow between the various entities comprising system **100** can be organized during game playas required by the specific game being played. For example, as described below, users can use headsets to communicate voice signals with each other. In such cases, the headset to headset communications can be established via temporary communication links **110** between two or more client devices. Alternatively, some or all of the communications flow can be established prior to game play. Communication flow is described in more detail below as well.

[0027] Users can use a variety of client devices to interface with game authorities **108** including desktop computers, laptop computers, handheld or palm computers, mobile communication devices, such as wireless telephones, Personal Digital Assistants (PDAs), and the like, as well as custom game consoles. In the example illustrated in FIG. **1**, client device **104** is a desktop computer. Depending on the embodiment, client device **104** can be configured to run the Windows® operating system or another operating system such as Linspire™. Client device **108** on the other hand is a custom game console, such as the XBOX® game console. Each client device comprises a game browser, which will be discussed in more detail below.

[0028] System **100** can be configured to provide a 3-Dimensional (3D) videogame environment presented, e.g., by game servers running on game authorities **108** to users via game browses running on the user's client devices. In one embodiment, the game servers running on game authorities **108** and the game browsers running on client devices **104** and **106** are not game specific, i.e., the game server and game browser can be generic with respect to the games. Accordingly, no customization of the game server and/or game browser is required to play different games. Moreover, as explained below, game developers developing games for system **100** do not need to be concerned with what type of client device the user, or users, are using. The game developer can simply develop one version of the game for system **100**, which can then be accessed and played by users using any type of client device.

[0029] Unlike conventional gaming systems, the game is stored on game authorities **108** as opposed to the client devices. Portions of the game environment can then be downloaded to each client devices as required. By only downloading portions of the game at any given time, the communications and processing bandwidth within the system can be conserved enabling the game to be hosted by a game authority **108**. In one embodiment, the approach used is similar to the structure of a web browser-web server combination. Websites comprise Hyper Text Markup Language (HTML) files that can be interpreted by a web browser.

[0030] When a website is visited using a web browser, the basic HTML files served by the web site to the browser contain "tags" that described how the "pages" associated with the web site are to be displayed on the user's computer. Some of these "tags" point to other resources, such as pictures, text and even links to other "pages," associated with the web site or with other web sites. In system **100**, when a game is accessed via a game server on a game authority **108**, a document is served to the game browser running on a client device. The document served by the game server can provide the game browser with information needed by the browser to conduct game play. The document, or game page, can act as a directory, or database, that contains a list of resources associated with the game being played. The resources can be images, executables, sound files, etc. associated with the game.

[0031] Different resources can be associated with different parts of the game. Thus, the directory can also inform as to what resources are associated with which portions of the game. As the player moves through, or progresses in, the game, the user's game browser can interpret the directory to determine what resources will be required in the near future. The game browser can be configured to communicate with the game server, running on the associated game authority, or authorities **108** to prepare to download, or access, the resources that will soon be needed.

[0032] The additional resources can be requested in the same manner as the game page. For example, a request for a resource can be sent to a Universal Resource Locator (URL) with the appropriate parameters for the requested resource in the header portion of the URL. Again, this is very similar to how web based activities are performed, e.g., this is very similar to "posting a form" in a web environment. Once the game server receives a request for a resource from a web browser, it can be configured to respond by sending the requested resource back to the game browser making the request.

[0033] The game browser can also be configured to "Cache" resources. Thus, when the game server receives a request for new resources, it can be configured to determine what resources already reside on the client device and only download resources not present on the client device.

[0034] Thus, in this manner, the user, or game player, can navigate through the game in a normal fashion; however, the majority of the game resources are actually stored on the associated game authority **108**. Only the resources that are immediately needed, and those that may be needed in the near future reside on the user's client device. Effectively, the interaction between the game browser on the user's client device and the game server on the associated game authority **108** maintains a "halo" of resources around the user as the user navigates, or progresses through the game. The game browser provides the user with the game play experience by interpreting and presenting the information extracted from the game server over network **102**, e.g., the Internet. But just as a web browser does not need prior knowledge of a web site to allow a user to view pages associated with the web site, the game browser can be configured such that it does not need any prior knowledge of a game, or game site, hosted by a game authority **108**.

[0035] FIG. **2** is a flow chart illustrating an example method for implementing a server based video gaming system from the perspective of a game browser running on a client device in accordance with one embodiment of the systems and methods described herein. The example method of FIG. **2** assumes that the user has already accessed a game server running on a single game authority **108**. In step **202**, the user can select a game to play. In many embodiments, game authority can be configured to host a plurality of games. Thus, there must be some mechanism by which the user can select a game. In one embodiment, the user first "arrives" in a "launch room." The "launch room" can comprise a plurality of doors, with each door being associated with a different game. The user can then move, via controls on their client device, through the door associated with the game they have chosen.

[0036] Going through the door can cause the game experience to begin. For example, The door through which the user moves in order to select a particular game can actually be a set of links to another "area," e.g., the starting zone for a particular game. Moving through the door can be the web equivalent of clicking on a web link. In other words, moving through the door can cause the game browser to send a URL request to the game server requesting the game page and/or system resources associated with the area of the game, such as the start zone, into which the user is moving. As explained in more detail below, videogame play can then comprise moving from area to area, or progressing through a game, which can be the equivalent of clicking on links in order to access more resources.

[0037] In step **204**, the game page associated with the game selected can be downloaded to the client device and interpreted by the game browser. As mentioned above, the game page can comprise a directory of resources required to implement the game play. In step **206**, those resources required to display the game environment inside the "active halo" around the user can then be downloaded and displayed on the display associated with, or interfaced with, the user's client device.

[0038] As mentioned above, the entire game never resides on the user's client device. Rather, only the area immediately surrounding the user is downloaded. As the user moves or progresses through the game, the game browser is monitoring the user's movements and/or progression and anticipating what resources will be required. The term "active halo" is intended to refer to that area of the game that is resident on the user's client device. The term "halo" is not intended to limit the active area around the user to a halo shape. The active area can actually comprise a circle, oval, square, etc. Nor is the term "active halo" intended to refer to just that area being displayed. It can also comprise resources and parts of the game not actively displayed, but resident on the client device. It should be clear that in order for game play to remain smooth and uninterrupted, more of the game than just that portion being displayed should be resident on the client device.

[0039] As the game progresses, the user's client device can be configured to determine whether there is movement, action, or progression associated with the game that requires, or soon will require, resources not resident on the client device. As an example, if the game is a role playing game, or an action game, where the user controls the movements of a character within the game, then new resources will be required as the user moves the character through the game. As will be understood, such games typically comprise a diverse geographic layout. Accordingly, the "active halo" surrounding the user's character, and resident on the user's client device, can comprise all the resources required to display the area immediately surrounding the user's character. As the user moves the character around within the game environment, new resources, e.g., models will need to be displayed or used. Thus, as the user moves the character around within the game environment, the client device can be configured to detect this movement, or activity, and update the display accordingly in step **210**.

[0040] Updating the display can comprise changing views as the user's character moves in different directions and displaying different aspects in the surrounding environment as the character moves from one place to the next. The client device can be configured, in step **212**, to inform the associated game server of the movement or activity so that the game server always has the most updated information related to the progression of the game. This can, for example, be important in multiplayer games, where the players reside at more than one remote location.

[0041] As the character moves within the game, new resources, i.e., resources not resident on the client device, will eventually be required. Thus, in step **214**, it can be determined if new resources are needed, e.g., based on the game page previously downloaded. If new resources are needed, then they can be requested and downloaded in step **216**. As stated above, the request can be in the form of a URL request. In certain embodiments, the client device can be configured to predict what resources will be needed and ask for them ahead of time. It will be appreciated that the resources should be resident on the client device before they are need in order maintain game flow, etc.

[0042] In step **218**, it can be determined whether there has been any change in the surrounding environment. For example, another character may have moved into sight, a car may have driven by, etc. If the environment has changed, then the display can be updated (step **210**). New resources may need to be downloaded as a result of the environment change. If so, then this can be determined in step **214** and the requisite resources downloaded in step **216**.

[0043] FIG. **3** is flow chart for implementing a server based video gaming system from the perspective of a game server running on a game authority in accordance with one embodiment of the systems and methods described herein. In step **302**, the game server can receive a game selection from a client device. In response to the game selection, the game server can be configured to download a game page in step **304** as described above. In step **306**, the game server can receive a request for new game resources. These game resources are associated with the portion of the game that will be resident on the client device. The rest of the game remains resident on the game authority. In step **308**, the requested resources can be sent to the client device.

[0044] Sending the resources comprises sending the resources and the scripts that tell the client device how to implement or use the resources. Thus, bandwidth can be conserved, because the data can be sent in a very compact way. In fact, often the client device will already have all the resources resident on the client device that the client device needs to implement the next portion of the game environment.

All that will need to be downloaded are the scripts necessary to make use of the resources to implement the next portion of the game environment.

[0045] In step **310**, the game server can receive updates from the client device. For example, the updates can be related to the movement or activity of a character within the game environment running on the user's client device. The game server can update the active version of the game in step **312** with this new information. Further, if it is a multiplayer game, where players are using more than one client device, e.g., at more than one location, to access the game server, then these other users' will need to be updated as to the changes in the game environment. Accordingly, the game server can determine in step **314**, whether it is a multiplayer environment, and update the other players in step **316**.

[0046] In step **318**, it can be determined whether there are any non-player controlled updates to the game environment. If so, then these updates can be provided to the users in step **316**. Non-player controlled changes can comprise changes generated by the game itself, i.e., the movement of game generated characters or components, etc.

[0047] Example resource types that can be associated with a game can include, spatial links, e.g., to the next game page in each possible direction. Specifications on other players currently viewing the current game page, i.e., information related to other players, or characters associated with another user that are in the same area of the game, can be included. Sound files can also be included. All of the sound files associated with a game can be referred to as a sound track. As explained below, there can even be multiple sound files, or tracks, associated with the same game. Models for the various components in the game can be included. Again, as explained below, multiple versions of the models can be included. Other resources can include, animations, textures, and control mappings, i.e., what key on the computer, or client device does something in the game. Again, the animations and textures can have multiple versions.

[0048] Aspects like rooms, forests, creatures, and even other players, or characters can be represented by textured models, which can be a combination of a 3D model with an accompanying texture.

[0049] Like the modem web, the game server and game site, i.e., the collection of game pages, are "dynamic," i.e., they change as time passes. The causes of such change can be varied. Some causes are simply a function of time, or a function of the game itself. Some are direct or indirect results of individual player's actions. Some causes are random, or dependent on outside monitoring of real world events. The varied causes of change that keeps the game environment dynamic help keep the games interesting.

[0050] By implementing the processes of FIGS. **2** and **3**, the game resources reside on the server with only a portion of the game residing on the client devices. As a result, a server based gaming environment can be implemented. As illustrated in FIG. **1**, and explained in more detail below, client to client communication can also be provided, as can authority to authority based communication, e.g., in multi-authority embodiments. Multiple game authorities can be used, e.g., for load balancing, redundancy, and to increase download speeds.

[0051] Referring back to FIG. **1**, a communication overview will be provided. There can be three basic types of communication links associated with a system **100** as configured in accordance with the systems and methods described herein. First, as illustrated, system **100** can comprise client-server communication links **112** and **116**. These links can, depending on the embodiment, be initiated by the client device or game authority; however, in most cases the client device will initiate connectivity with a game authority **108**. Links **112** and **116** can be used by the game browser to access the game page, request and download resources, provide update information to the game server, etc.

[0052] Second, system **100** can comprise client-client links **110**. These links can allow for direct communication between client device, e.g., client devices **104** and **106**. In many multiplayer games, communication between user's can be essential, e.g., were multiple players are on the same team. These communications can be relayed via game authority **108**; however, in many instances there is no need to burden game authority **108** with the client to client communications. Thus, for example, link **110** can comprise VoIP communication links over which user's can communicate with each other, e.g., using headsets interfaced with their respective client devices. In certain embodiments, transcripts, or logs, of the communication between users, or client devices, can be stored by one or more of the client devices involved. The stored logs can then be uploaded to the server at a later time.

[0053] Third, system **100** can comprise inter-server communication links **114**. Link **114** can comprise LAN links and/or WAN links depending on the embodiment. Link **114** allows multiple game authorities, or game servers, to synchronize with each other. As mentioned, multiple authorities can be used, e.g., for redundancy, load sharing, or faster download capability.

[0054] a. The Game Server

[0055] FIG. **4** is a diagram illustrating an exemplary game server **400** configured to run on a game authority **108** in accordance with one embodiment of the systems and methods described herein. As illustrated, game server **400** comprises communication processor **402** and game play engine **404**, and is interfaced with data repository **408**.

[0056] Communication processor **402** can be configured to process communication traffic between game server **400** and other game servers as well as between game server **400** and client devices. Communication processor **402** can, for example, be configured to queue requests for information from client devices, or other game servers, and passes back the appropriate responses from the game play engine **404**.

[0057] Game play engine **404** can be configured to process game scripts **406** associated with various games loaded onto the associated game authority **108**. The games, or game scripts **404** can be provided by game developers as described in more detail below. Game scripts **406** control the interaction between user's, game servers, and combinations thereof. Game play engine **404** can be configured to service requests for information by pulling the requested information from, e.g., the data repository **408**. Game play engine **404** can be further configured to record information sent back from the client devices regarding the associated user's various exploits within the games.

[0058] Game play engine **404** can be configured to run all game scripts **406** provided by the various game developers, and is the server side of the software which determines what players will be "managed" for interaction, etc. The rest of this interaction is handled by a processor in the client device. Game play engine **404** can be where all of the "universal gameplay" tasks are handled. Rankings, alliances, and accumulated “treasures” can all be handled here. Of course, the final determination of what resides where rests with the game developer.

[0059] As mentioned above, game play engine **404** can be configured to simply download the resources and scripts needed to implement a current portion of the game environment to the client device. Often, the client device will have all of the resources necessary to implement the next portion of the game environment already cached on the device. Thus, all the game play engine needs to do is serve the scripts that tell the game browser running on the client device how to use the resources to implement the next portion of the game environment. Before downloading new resources, game play engine **404** can be configured to determine what resources the client device has cached on the device and only serve the resources and scripts not already resident on the client device.

[0060] Data repository **408** is where all of the resources **410**, e.g., models, their textures, sounds, music control specifications, links, etc., are stored. Everything there is to know about a game should be stored in data repository **408**. For example, all of the motion rules, which determine how an object responds to player inputs on the client device can be stored in data repository **408**. Data repository **408** can also be configured to store “control templates,” which can translate the controls the game needs, to what the player’s client device is equipped with, and “extended object trajectories,” which can help to determine what will happen to an object in motion through multiple areas. These extended object trajectories can be used to allow a game browser to predict which will be needed and when they should be requested.

[0061] Depending on the embodiment, data repository **408** can be a database or file structure. A database is much easier to maintain, copy, synchronize with other servers, create mirrors, etc. In fact, there are many advantages to the use of a database; however, cost can be a disadvantage. The same tasks can, depending on the embodiment, be accomplished with simple files and data structure; however, such implementations tend to be much less standardized and much more system dependent. Accordingly, the appropriate tradeoffs should be made on an independent basis between cost and ease of implementation.

[0062] Game servers **400** are capable of communication with each other as well as the client devices. This allows a game hosted on multiple game servers **400** to have some commonality. One game can be on many servers **400**, or one server **400** can have many games. For example, ten servers **400** can be home to 10% of each of 10 completely different games. Any combination is possible, which provides a great deal of flexibility in resource allocation. If a game is extremely popular, dedicated servers **400** can be used to handle the load; however, if there are 10 mildly popular games, the former scenario is probably a more efficient use of resources.

[0063] b. The Client Device

[0064] FIG. **5** is a diagram illustrating an example client device **501** comprising a game browser **500** configured in accordance with one embodiment of the systems and methods described herein. As can be seen, game browser **500** comprises communication processor **502**, game processor **506**, sound processor **508**, graphics engine **510**, and is interfaced with data repository **504**.

[0065] Communication Processor **502** can be configured to process all of the data coming from gamer servers and other client devices. Communication processor **502** can be configured to pass advance requests on to data repository **504**, while more immediate information can be sent to game processor **506**. Advance requests can, for example, be data that game browser **500** has determined will be needed in the near future, such as the model for the next area in the current direction of travel, while the more immediate information can be data and/or changes regarding the current position within the game environment.

[0066] Game processor **506** can be configured to receive information about how the controls and display will be formatted, how all of the objects in the given area react to various other objects and the player, and of course, how the player interacts with a particular scene. While communication processor **506** can be running in the background, e.g., feeding data repository **504**, game processor **506** can be running in the foreground controlling graphics engine **510**, sound processor **508**, and of course interpreting the player’s responses. Game processor **506** can be configured to use data stored in data repository **504**, make its own entries in the data repository **504** for future reference, and sends back reports and requests to communications process **502**.

[0067] Graphics engine **510** can be configured to draw and animate the current scene on the display associated with client device **501**. Game processor **502** can be configured to instruct graphics processor **510** as to which models, textures, and positions to use, but graphics processor **510** has the complex task of actually rendering the effect, e.g., in 3D on the client device display.

[0068] Sound processor **508** can be configured to play the music track, sound effects, and any other incoming or outgoing audio, such as from a headset-to-headset device.

[0069] Data repository **504** can, depending on the embodiment, be a database established on client device **501**. Thus, it is preferable that each client device within system **100** comprise some sort of mass storage. The actual data storage requirements can vary by game, and can also be related to the level of support given to a certain type of device. For instance, a PC can have a given 3D model’s texture represented at 1024×768 pixels and 65535 color depth, while a PCS phone playing the same game can only be 102×77 with a 16 color depth.

2. Additional Features

[0070] The systems and methods described above enable a server based game environment, because only portions of the game are downloaded to the client devices. Thus, bandwidth is not an issue and the game can be played without any interruption of game flow or game experience. The game server can also update one player with information related to another player in a quick and easy manner. The systems and methods described above can, depending on the embodiment, lead to other features and advantages as described below. It will be clear that the systems and methods described above can provide other features and advantages as well and that the features and advantages discussed below are simply exemplary of the enhanced performance that can be provided by the systems and methods described herein.

[0071] a. Multi-Track Games

[0072] Because a game authority **108** stores the resources **410** and scripts **406** required for game play, multiple versions of the same game can be stored and made available for users. For example, if the game designer chooses to do so, various versions, e.g., mild, mature, and violent, of a game can be designed and loaded on game authority **108**. The user can then be given the option of which version of the game the user wants to play. Alternatively, the game developer can produce multiple versions, where certain versions have enhanced or added content. The user can then, for example, be given the option of which version they want to play. If the user selects a version with enhanced or additional content, the user can be charged more. In another embodiment, different language versions of a game can be developed. The user can then select the language version they desire when selecting a game. Different language versions can comprise simply providing different sound tracks and audio effects as mentioned below.

[0073] Conventionally, the game developer is forced to design games at great cost to the developer and with very little guarantee of any return. This makes it very difficult from a practical standpoint for the developer to design more than one version of the game. As a result, games are designed to appeal to the broadest market, but in so doing certain section of the market may be turned off by graphic content or high prices. By combining the storage and access capability of the systems and methods described above, with the development processes described below, multi-track capability is made practical. This can provide access to sections of the market previously inaccessible for one reason or another, by providing multiple versions of the game or the content/resource therein.

[0074] The multi-track capability can also apply to the sound track and audio effects as well as to the graphical content of the games.

[0075] b. Parental Control

[0076] A consequence of the multi-track capability described is the ability to provide easy to use and implement parental control. For example, at least two version of a game can be stored on game authority **108**: a child friendly version; and an adult version. When a user accesses a game, they can indicate which version they want. Moreover, as part of the users subscription, they can indicate a parental control election that can govern what version can be accessed. Subscriptions are described in more detail below.

[0077] Accordingly, the systems and methods can provide increased parental control and more options, e.g., to families.

[0078] c. Piracy Protection

[0079] Piracy is a problem that plagues the game industry. Piracy is the unauthorized copying and distribution of a game. Copy protection techniques are well known and widely used; however, they have proven ineffective at preventing piracy. But because a whole version of game is never on any client device at anyone time, games hosted by game authorities **108** cannot be easily copied. In order to effectively copy a game, a user would have to navigate through the whole game, i.e., every option, path, experience, etc., while recording all of the data at the same time. This presents a complex problem for a would be pirate that cannot be easily overcome. As mentioned above, encryption can also be employed to help prevent the unauthorized access and copying of game content.

[0080] d. Player Rating

[0081] System **100** can be configured to allow game players to be rated as to their skill level for various games. Skill level ratings can be useful for a variety of reasons. For example, a skill level rating can be useful for the game player to rate how good they are at a game relative to other players and/or to track their improvement in playing a given game. Additionally, it is envisioned that leagues will become popular, where players join to play against each other, or the game, alone or in teams. In this context, skill level ratings can be used to select teams, pair opponents, seed competitions, etc.

[0082] Accordingly, game server **400** can further comprise a player rating engine that can be configured to gather data related to a players ability in a variety of ways in order to produce an accurate and measurable score based on their demonstrated ability.

[0083] FIG. **6** is a flow chart illustrating an example method for determining a players skill level from a game server perspective in accordance with one embodiment of the systems and methods described herein. In the example of FIG. **6**, the user can elect to have their ability in relation to a particular game evaluated and rated based on standardized simulations. The standardized simulation can be designed to gauge a user's ability level with regard to key skills needed to play the associated game effectively.

[0084] Thus, in step **602**, a game server **400** receives an election to be tested from the user via the user's client device. As explained above, the election can actually comprise a request for the resources need to present various test simulations on the user's client device. In step **604**, a test simulation can be presented to the user. This can comprise game server **400** sending the appropriate resources and scripts to the user's client device. The simulation that a user is run through can be dependent upon the game of choice. For example, if the user desires to achieve a skill level rating in a first person shooter environment, they can be presented a detailed and rigorous testing environment that focuses on the skills and reflexes that are required to compete at a high level in such an environment.

[0085] In step **606**, data related to the users results or skill can be stored, e.g., in data repository **408**. In step **608**, game server **400** can determine if the testing, or evaluation is complete. If the testing is not complete, then further simulations can be served to the user's client device. Once all simulations are complete, then the player's skill level can be determined, e.g., based on the data stored in step **606**. For example, the results obtained for each simulation or test environment can be combined to produce a score that is indicative of the user's ability in relation to a certain game.

[0086] In certain embodiments, the score can actually provide depth that can allow the user to determine what skills need to improve in order to achieve a higher score. Moreover, the score can be generalized so that it provides an indication as to how the user can perform in relation to a certain type, or genre of game.

[0087] A report of the user's results and skill rating can be produced in step **612**. The report can then be provided to the end user, e.g., via a display on the associated client device. Alternatively, the report can be made available via the Internet, World Wide Web, email, etc. The report can also be made available to the gaming public, or a portion thereof, for evaluation, recruitment, rating, etc.

[0088] For example, in the first person shooter environment context, the user can receive detailed stats on their areas of fault and mastery. They can, e.g., receive a packaged electronically generated report that contains a thorough analysis of all of the key categories of first person shooters. As an example, the user can receive a score that shows how they ranked in areas such as team support, aim/accuracy/hit %, dodging ability, ammunition usage, tactics/strategy, etc.

[0089] Based on the reporting statistics the user can be provided with a detailed view of how they look from a graphical perspective, e.g., via a radar graph. By showing the optimal test score on the outer portion of the radar graph, the user can see how they can improve in the area of focus. For example, in the example radar graph of FIG. **7**, several skills are displayed around the outer edge of the graph. The lighter shaded portion illustrates the results that a model player would receive for the illustrated skills. The inner, darker shaded area illustrates the user's tested results. Thus, the user can see where he needs to improve in order to achieve the model, or average, or highest level or whatever the comparison may be. In this example, it is clear that the system has identified AIM/Accuracy as the area that needs the most improvement of the user's skill set.

[0090] In another embodiment, data from actual game play can be recorded and used alone or in combination with simulation data described above, to determine the user's skill rating. For example, raw data obtained during game play can be filtered to retrieve data indicative of a player's ability in key areas. The filtered data can then be used to generate a skill level rating, e.g., by generating a score as described above. The skill rating data can also be reported to the user as described above.

[0091] e. Virtual Camera Views

[0092] In certain embodiments, a virtual camera system can be incorporated into the game environment that can record game play from various views live. This can, for example, be used to enable gamers to view battles in real-time as well as in replay. Features such as slow motion, instant replay and embedded view can all be provided. Embedded view can allow viewers to view the action and replays from that of any player in the first person, i.e., enable a non game player to see what a player sees during the game. For example, players can access the recorded views, e.g., using their client devices, and replay the game from various views. Additionally, in certain embodiments, non-players can access a game authority **108** and view the replays, or live action, from one or more camera views. The non-players can access the recorded views using a client device as describe above, or can simply use a computer to access the a game authority **108**, e.g., via the Web. Alternatively, depending on the embodiment, the recorded views can be broadcast to viewers, e.g., who have signed up to receive such broadcasts. These broadcasts can be over Internet, cable, conventional broadcast media, etc.

[0093] Accordingly, each game can comprise "cameras" in various locations. These cameras are algorithmically generated and can be "placed" on the players, in fixed locations, or in movable locations. For example, a "cameraman" can be inserted into the game and associated with a "camera." The "cameraman" can then roam around the game independently recording what he sees. Depending on the embodiment, such a "cameraman" is not visible to the actual game players.

[0094] An interactive player rating module can also be included within a game server **400**. Such a module can allow viewers of the match to give insightful rating and commentary to specific players. For example, each player can be associated with a player profile. This profile can, for example, contain official ratings as well as viewer ratings and commentary, which can be used to measure the popularity of a certain user for his in game tactics and techniques.

[0095] In certain embodiments, game server **400** can also comprise an interactive commentator module. For example, upon the viewing of a game, a viewer can listen to active commentary that can be embedded via a media module. The media module can, for example, allow third parties to launch insert and record their own webcasts and play by play while the game is being played. This can allow multiple webcasters the ability to comment on the play of the game at the same time. This technique will allow fans to choose the online media outlet/commentators that they like best when viewing the matches.

[0096] Certain embodiments also comprise an interactive cam casino module. A cam casino module can be an add on component of the virtual camera capability described above. This module can enable viewers to watch and wager on the games being played and the performance of the player within the game.

3. Game Development

[0097] The systems and methods described above can lead to several advantages for the game developer, because a game authority **108** offers the developer a platform from which they can reach more users, with lower development costs, and even a mechanism by which they can raise funds to cover operating costs during development. Often game developers find it very difficult to generate the funds needed to develop games. Moreover, if they are developing games for a customized, or proprietary client device, or platform, then they must first get the approval of the device manufacturer. Otherwise, they risk being shut out of their target market, once the game is developed.

[0098] What often occurs is that the developer will get an agreement up front from a device manufacturer that pays the developer money prior to developing the product. The funds pay for development of the product, but typically cannot be used to pay other operating cost for the developer. Further, the amount paid is typically based on a projected amount of sales. If the game does not sell enough, then the game developer may be required to refund some of the money. Conversely, if the game is a big success with greater demand than anticipated, the developer does not necessarily get more money. In fact, since it is likely that only the projected number of games were manufactured and put on shelves, it is likely that this extra demand will go unmet.

[0099] System **100** can be configured, however, to significantly reduce the risks and costs associated with development for the game developer. For example, a game server **400** can be configured to provide a game developer with full access to server resources via game development scripts. The scripts can be used to develop games using a platform agnostic process. In other words, the developer can develop one version of the game that can then be accessed and played using a plurality of different types of client devices. Adaptations, for the specific controls and capabilities associated with a given platform can be handled using a short, platform dependent translation script. The translation script can be developed by the game developer, the platform developer, or even a third party. The translation scripts can be passed from game server **400** to a game browser **500**.

[0100] The game development scripts allow the game developer the ability to develop games quickly, for a variety of platforms, without being bogged down with engine specific code. The game development scripts can be implemented on both the game server and the game browser, which allows the load of the game to be shared between the two. Not only does this potentially reduce system bottlenecks, but it also allows the developer to decide where certain elements should be stored, i.e., on a game authority **108** or client device.

[0101] Additionally, a developer can develop a demo version of a game, e.g., just a few levels or areas, and post it to a game authority **108**. Users can then be allowed to access the demo. This can provide feedback to the game developer before he invests more time and money to complete the development. In other words, if a lot of users download the demo, then it is clearly worth continuing with the development. In fact, users can even be charged a fee to access demos, which can help to fund the rest of he development.

[0102] Another advantage of system **100** from the game developer's point of view can be the elimination of the need for a publisher. Conventionally, publishers are the ones who produces the packaging, labels, instructions, and disk associated with a game that is to be sold to the public. But since games in system **100** are simply uploaded to a game authority **100**, the need for the publisher is eliminated.

[0103] Similarly, marketing can be made easier. For example, once the game is uploaded, marketing can be handled through game authority **100** or a related system. As explained below, users can have subscriptions to access games on game authority **108**. As part of the service, information about new games can be pushed out to the users, or made available, in order to generate interests in new games. The demo process described above is also a form of marketing.

[0104] System **100** can also be configured to perform testing and validation of the games. In fact, the demo process can comprise part of the test and validation process. Separate test and validation capabilities and feedback can also be incorporated into game authority **108**.

[0105] Accordingly, the process of developing and bringing a game to the public can be made shorter, less expensive, and more profitable for the game developer using the systems and methods described herein.

4. Subscription Model

[0106] As mentioned, users can pay a subscription to access games on a game authority **108**. It should be noted that access can also be based on a fee per transaction basis or even for free depending on the embodiment and/or the situation. But in one embodiment, users can be offered a certain base subscription price for access to a certain number of games for a certain period of time. For example, a base subscription can provide unlimited access to three games for one month. If the user would like more games or more time, then the subscription fee can be raised accordingly.

[0107] The game developer can then be paid a royalty based on the subscriptions collected.

[0108] For example, in one embodiment, the game developer is compensated via a royalty that is determined based on the popularity of the developer's game. The popularity of a game can be determined based on how many subscriptions it is included in. Further, royalties can also be paid to certain platform developers. Again the royalty can be based on the number of users using a given platform to access game authority **108**.

[0109] Subscriptions prices can also vary as other services are added. For example, the user can pay in increased fee for access to demos, skill rating services, virtual camera services, and other features described above. These additional service fees can generate more revenue for the game developer and/or platform developer, depending on the embodiment and the nature of the service associated with the increased fee.

[0110] While certain embodiments of the inventions have been described above, it will be understood that the embodiments described are by way of example only. Accordingly, the inventions should not be limited based on the described embodiments. Rather, the scope of the inventions described herein should only be limited in light of the claims that follow when taken in conjunction with the above description and accompanying drawings.

What is claimed:

1. A client device in a server based videogame system, the client device comprising:

a data repository configured to store portions of resources and scripts required to implement a game environment associated with a videogame being played using the client device; and

a game browser interfaced with the data repository, the game browser configured to manage the resources and scripts in order to implement a portion of the game environment and to request more resources and scripts as required to implement other portions of the game environment.

* * * * *