US 20130190094A1

(19) **United States**

(12) **Patent Application Publication**

Ronen et al.

(10) **Pub. No.: US 2013/0190094 A1**

(43) **Pub. Date: Jul. 25, 2013**

(54) **ASYNCHRONOUS TEAM-BASED GAME**

(76) Inventors: **Eyal Ronen**, Ramat Hasharon (IL); **Doron Nir**, Tel Aviv (IL)

(21) Appl. No.: **13/353,779**

(22) Filed: **Jan. 19, 2012**

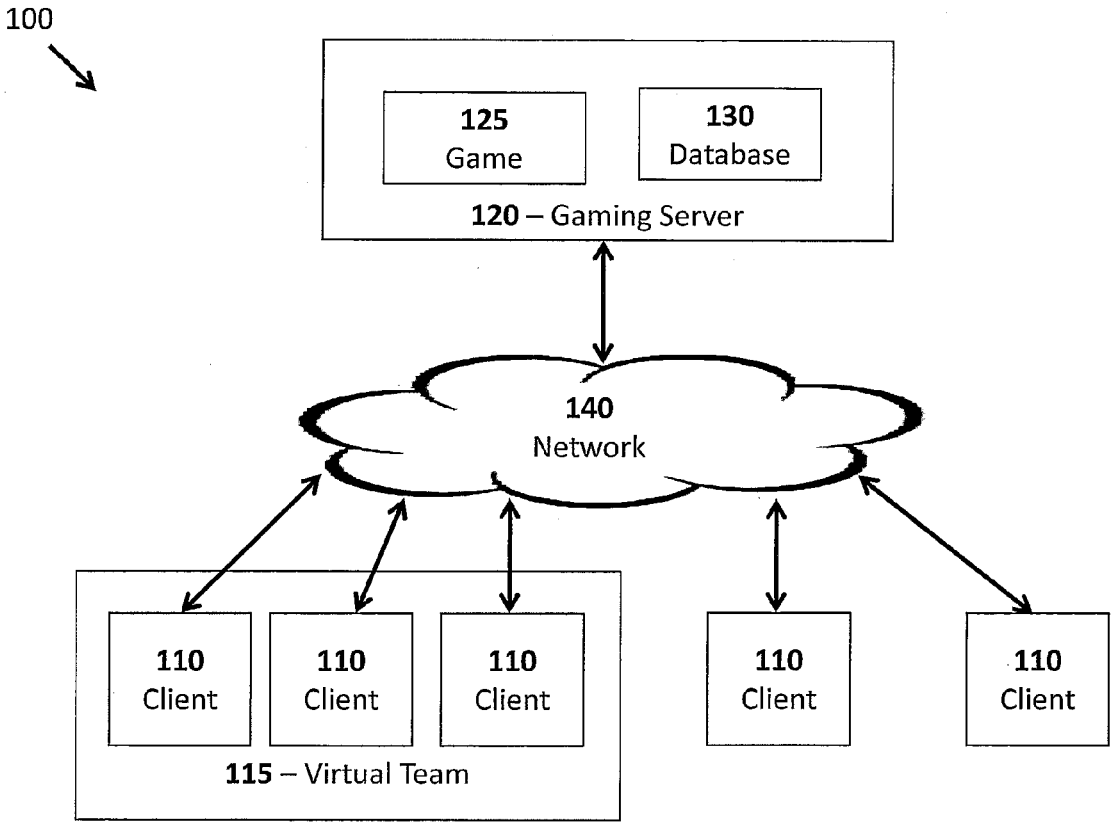
Publication Classification

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

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

(57) **ABSTRACT**

An asynchronous, multiplayer gaming platform is disclosed which provides for a collaborative team effort on a shared canvas. Shared elements (e.g., a shared canvas, shared object or a shared inventory) of a game are stored on a computer readable storage medium that is accessible to the players of the team over the network. Data is received from a first player of the team which is indicative of gaming activities modifying the shared elements. The shared elements are updated with the data received from the first player to generate an updated game state. Update data representing the updated game state is transmitted to at least one other player on the team.



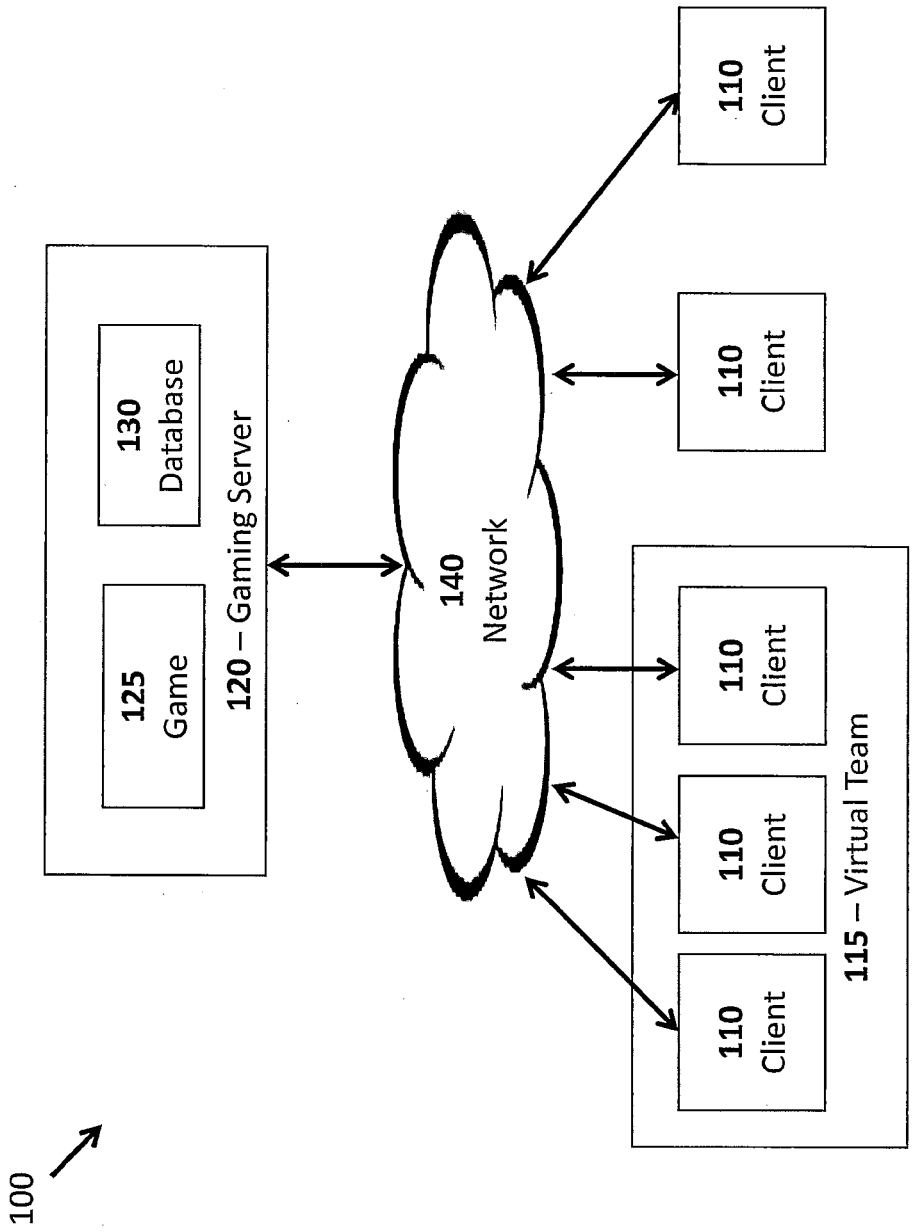

100
120 – Gaming Server
125 Game
130 Database
140 Network
110 Client
110 Client
110 Client
110 Client
110 Client
115 – Virtual Team

Figure 1

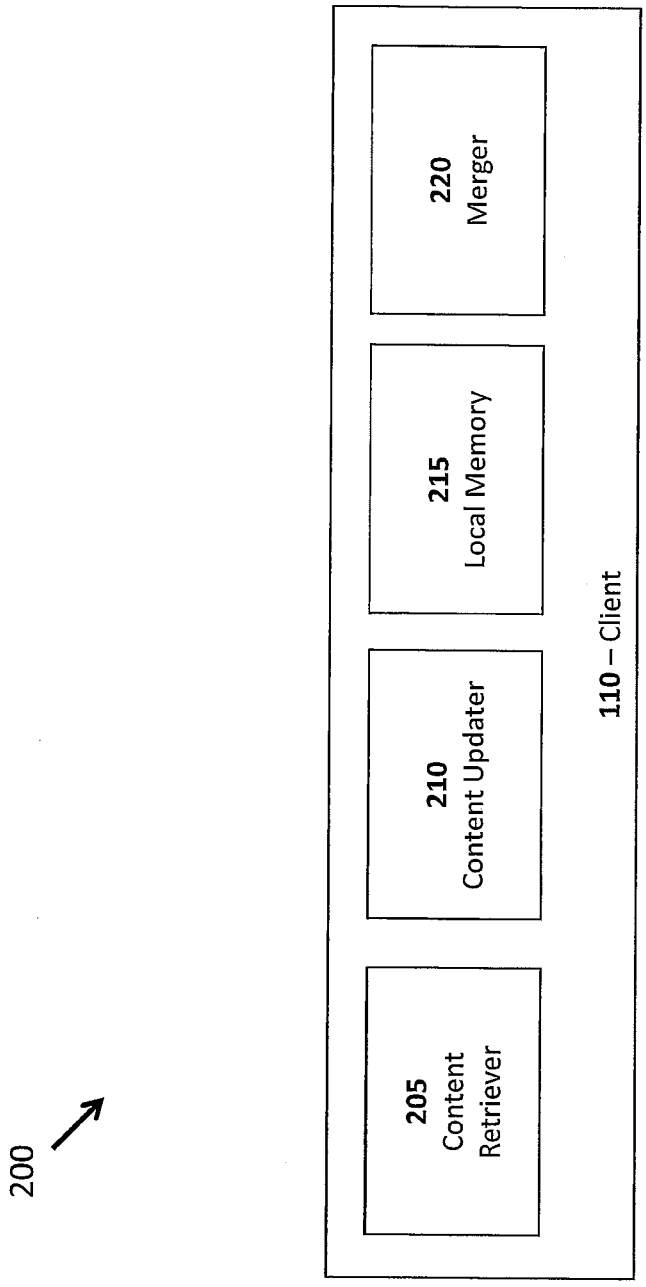
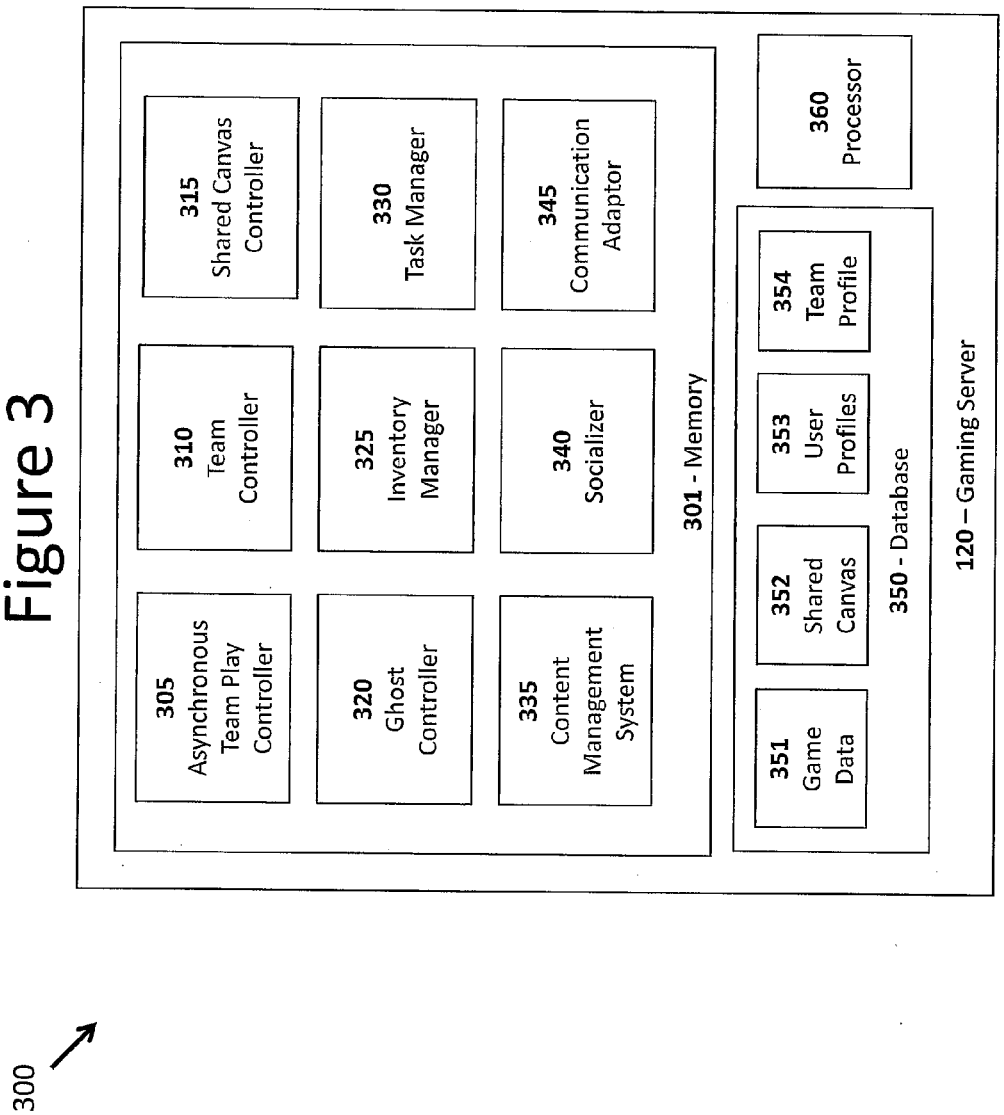

200
205
Content
Retriever
210
Content Updater
215
Local Memory
220
Merger
110 – Client

Figure 2

Figure 3


300
120 – Gaming Server
301 - Memory
305 Asynchronous Team Play Controller
310 Team Controller
315 Shared Canvas Controller
320 Ghost Controller
325 Inventory Manager
330 Task Manager
335 Content Management System
340 Socializer
345 Communication Adaptor
350 - Database
351 Game Data
352 Shared Canvas
353 User Profiles
354 Team Profile
360 Processor

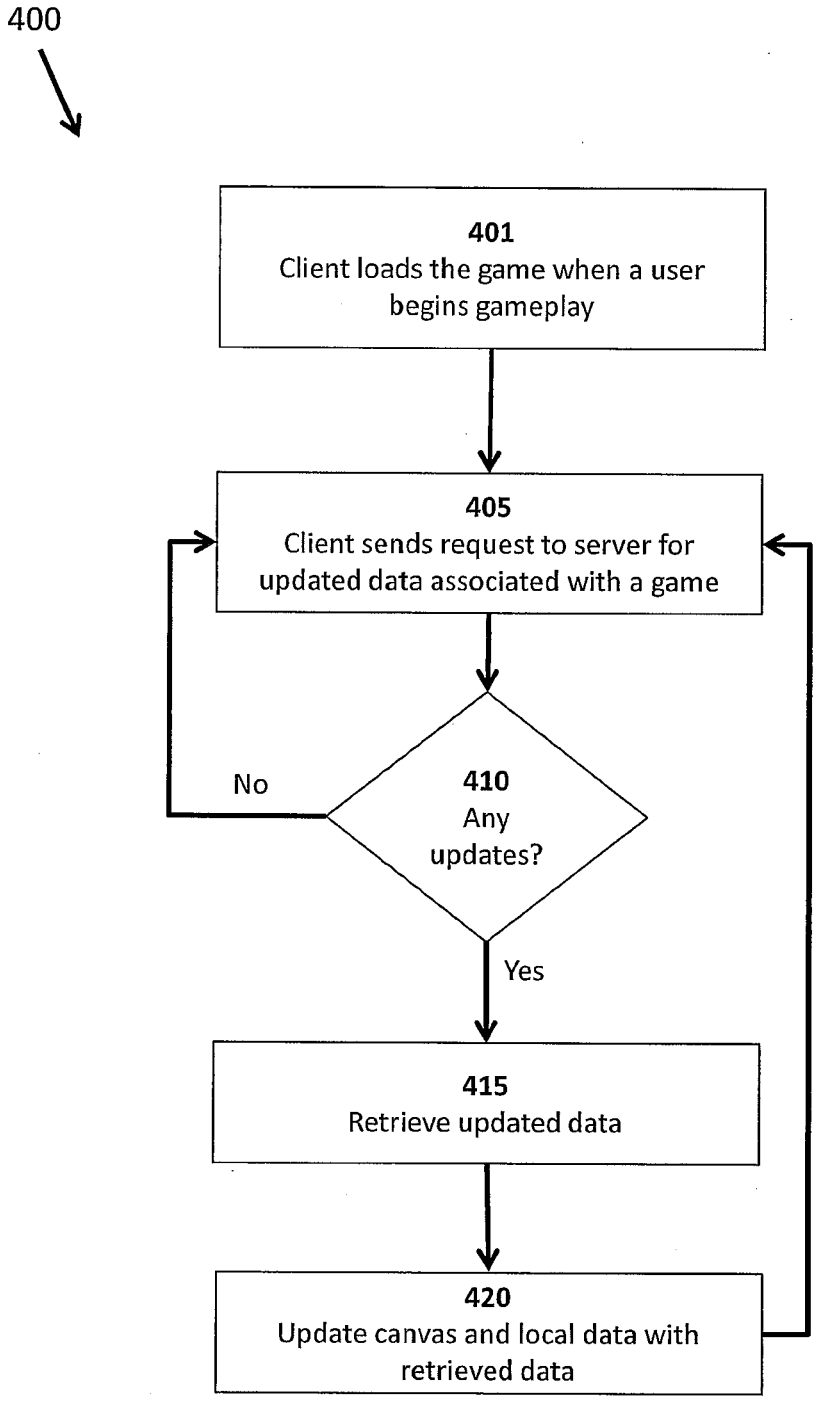

400
401
Client loads the game when a user begins gameplay
405
Client sends request to server for updated data associated with a game
410
Any updates?
No
Yes
415
Retrieve updated data
420
Update canvas and local data with retrieved data

Figure 4

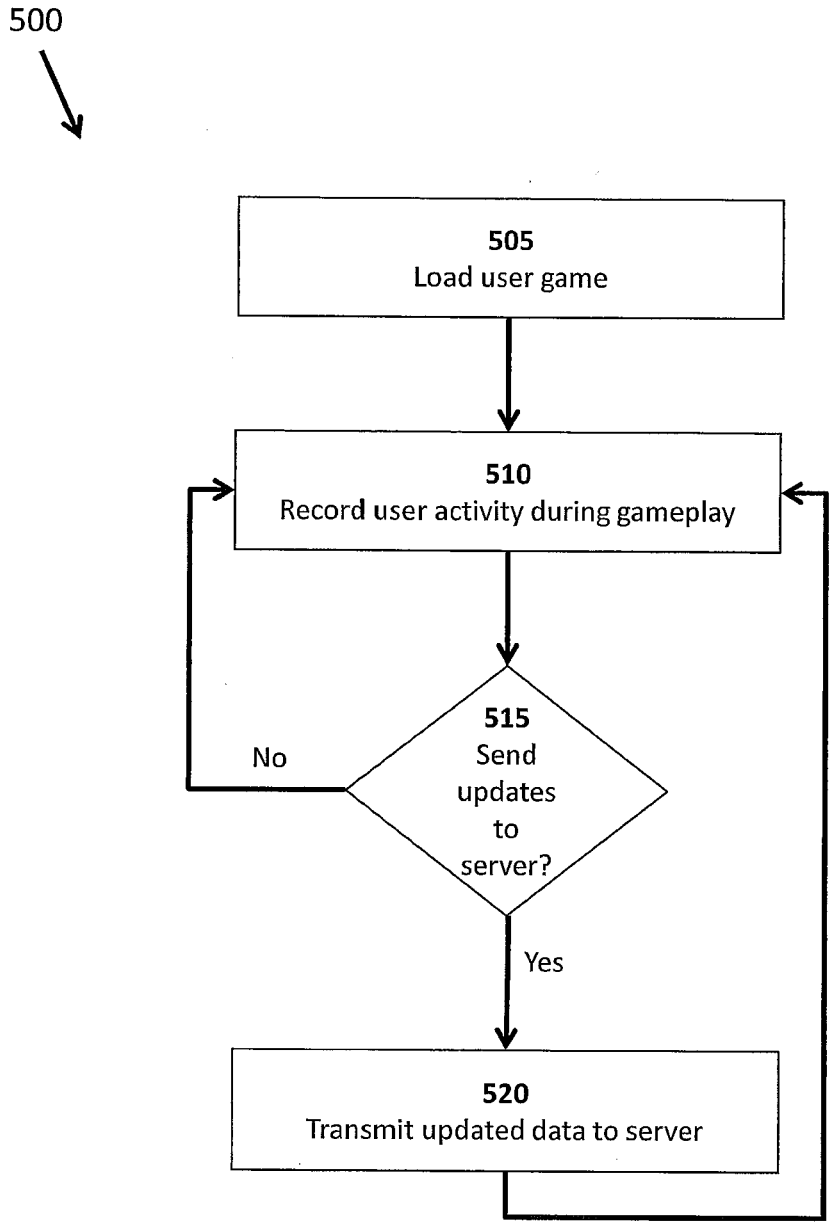

500
505
Load user game
510
Record user activity during gameplay
515
Send updates to server?
No
Yes
520
Transmit updated data to server

Figure 5

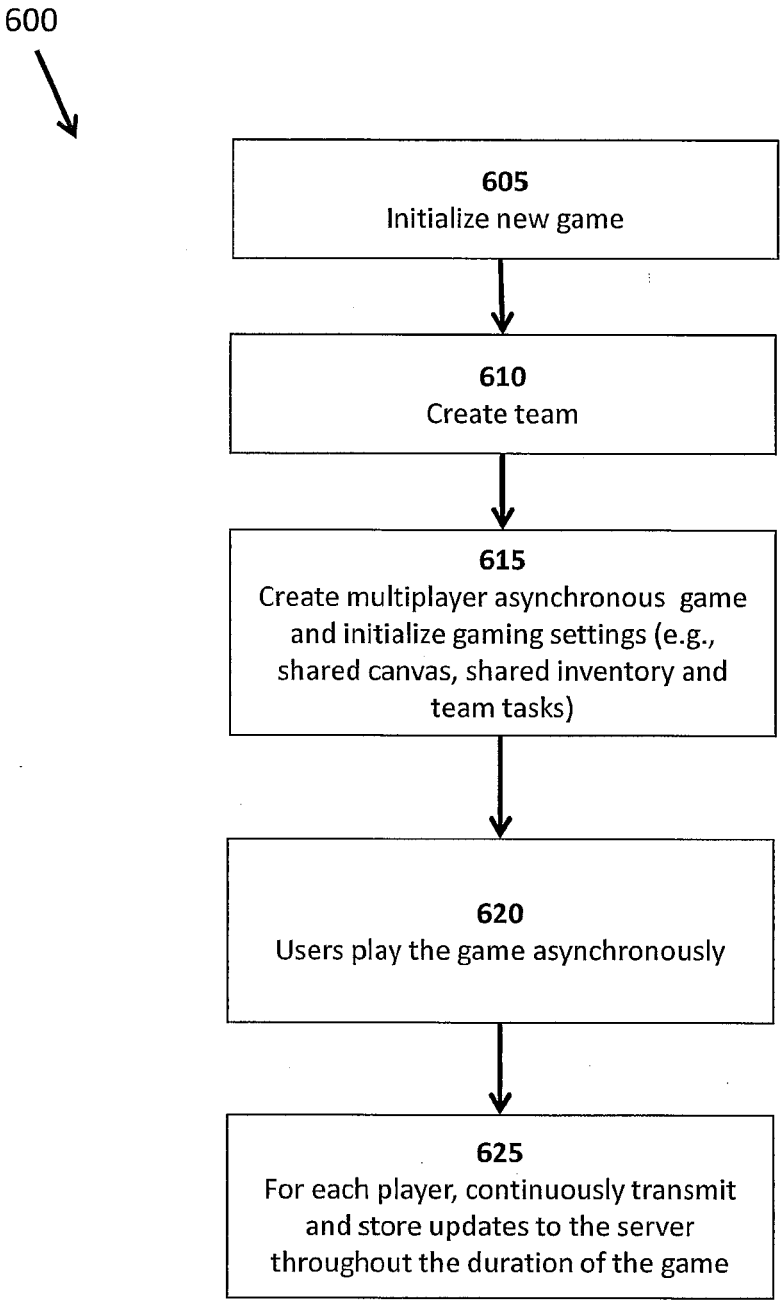

600
605
Initialize new game
610
Create team
615
Create multiplayer asynchronous game and initialize gaming settings (e.g., shared canvas, shared inventory and team tasks)
620
Users play the game asynchronously
625
For each player, continuously transmit and store updates to the server throughout the duration of the game

Figure 6

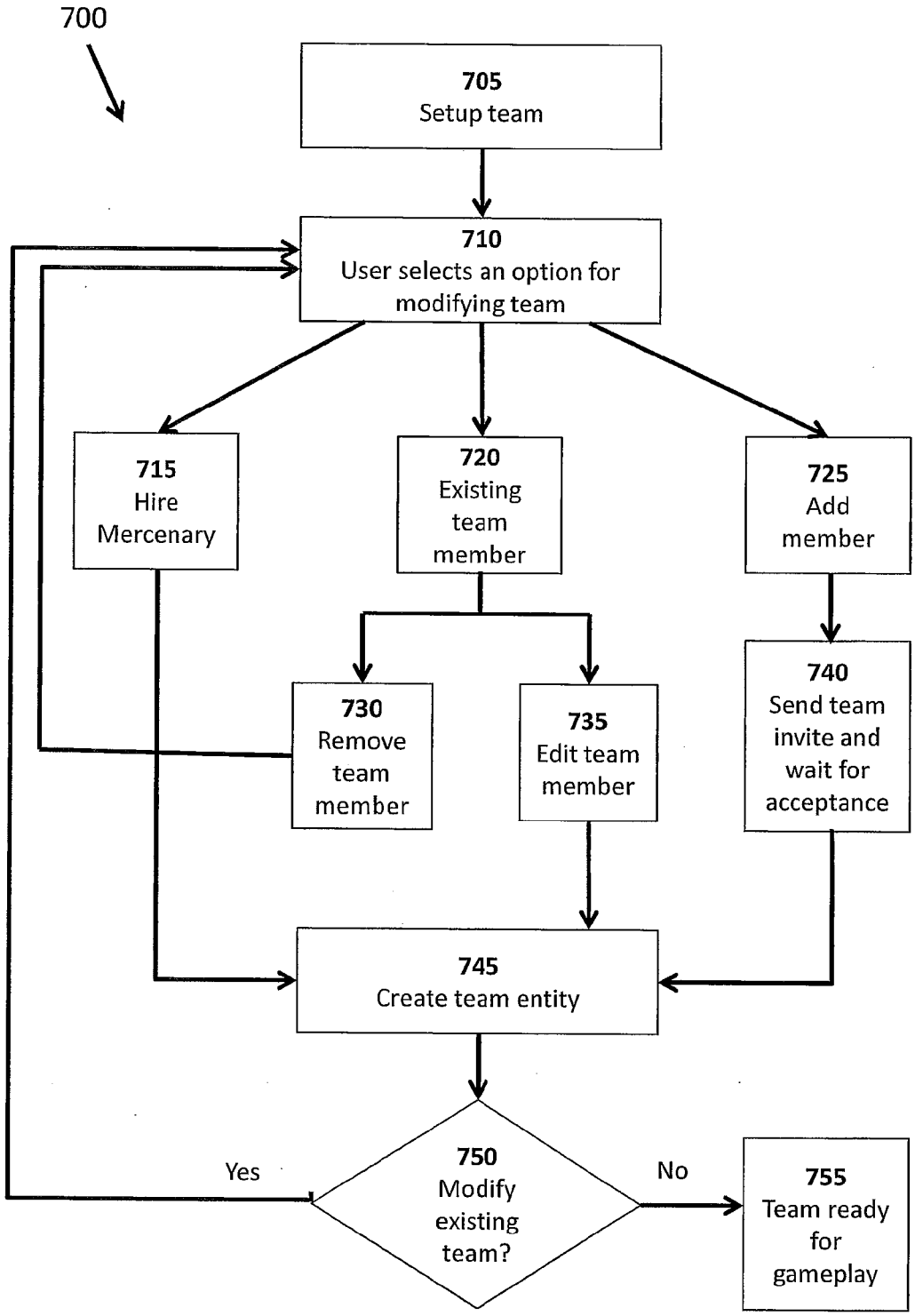

700
705
Setup team
710
User selects an option for modifying team
715
Hire Mercenary
720
Existing team member
725
Add member
730
Remove team member
735
Edit team member
740
Send team invite and wait for acceptance
745
Create team entity
750
Modify existing team?
Yes
No
755
Team ready for gameplay

Figure 7

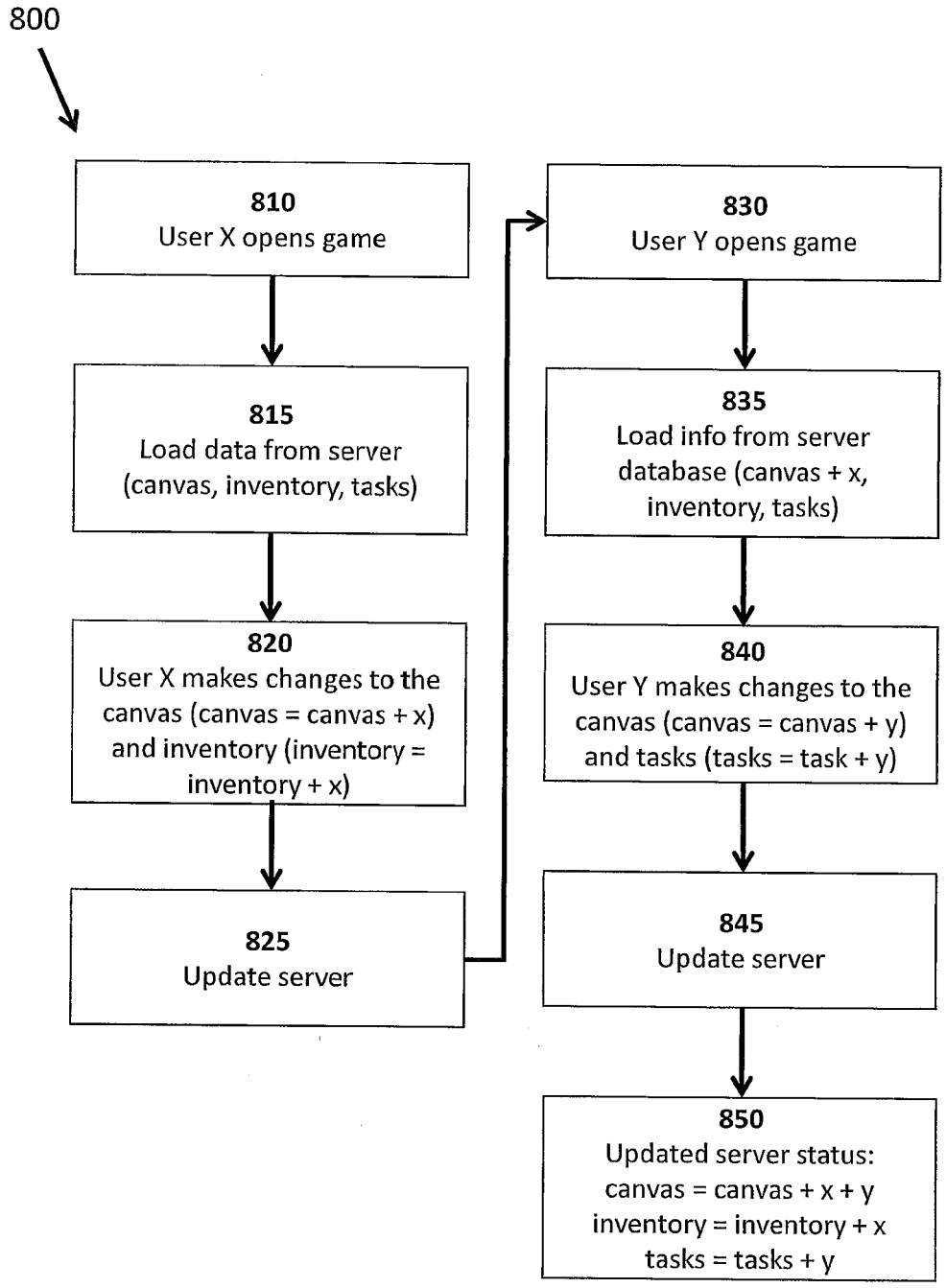

800
810
User X opens game
815
Load data from server (canvas, inventory, tasks)
820
User X makes changes to the canvas (canvas = canvas + x) and inventory (inventory = inventory + x)
825
Update server
830
User Y opens game
835
Load info from server database (canvas + x, inventory, tasks)
840
User Y makes changes to the canvas (canvas = canvas + y) and tasks (tasks = task + y)
845
Update server
850
Updated server status:
canvas = canvas + x + y
inventory = inventory + x
tasks = tasks + y

Figure 8

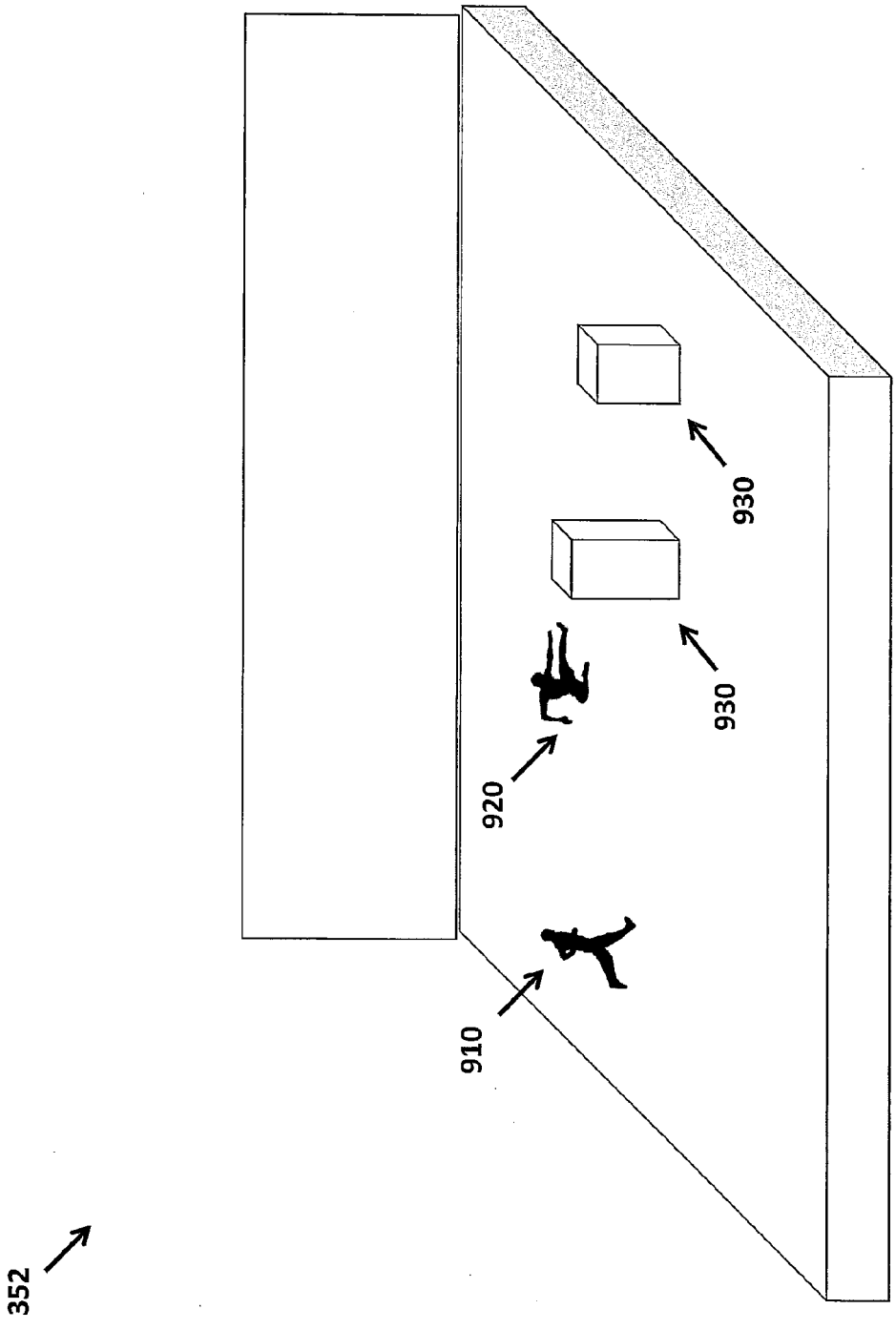


Figure 9

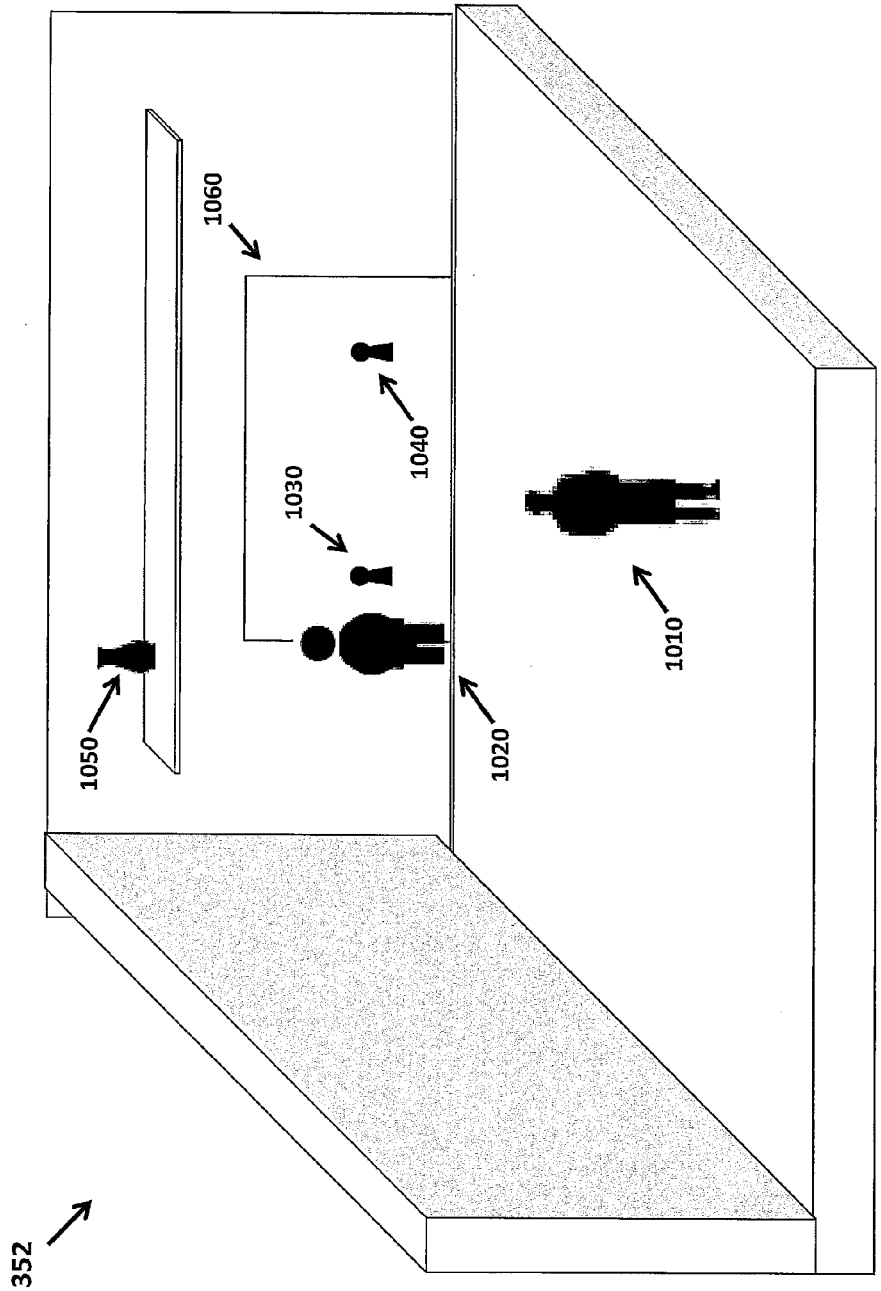


Figure 10

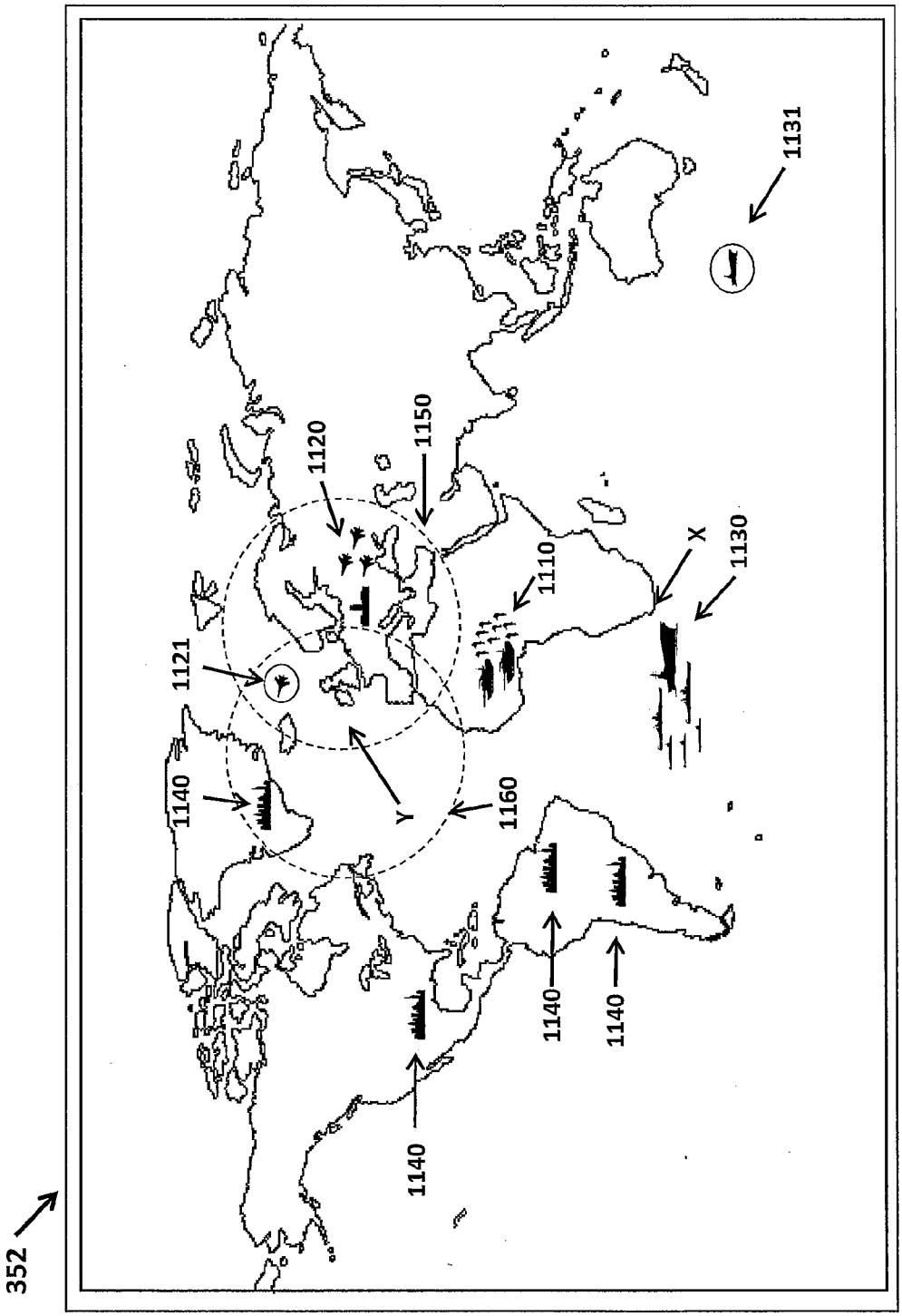


Figure 11

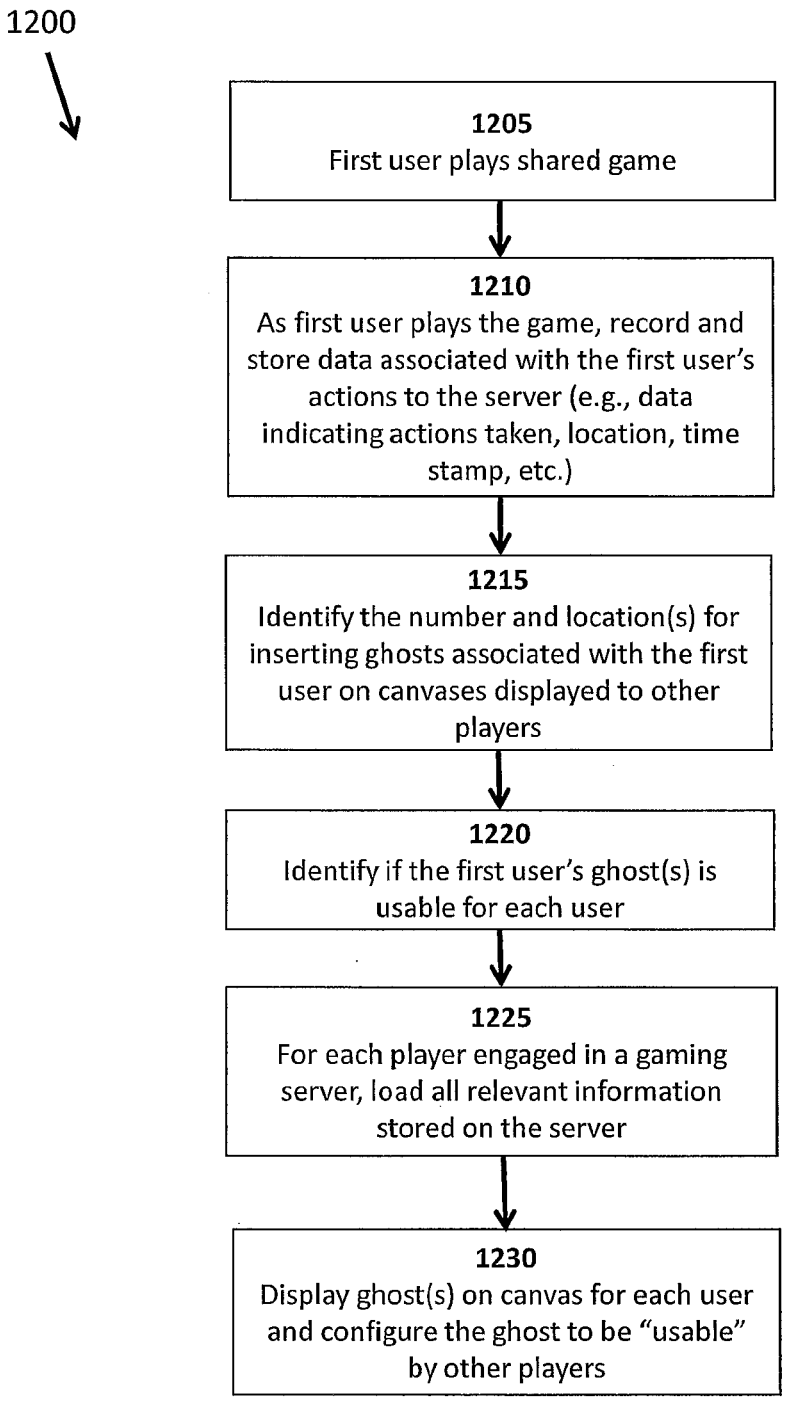

1200
1205
First user plays shared game
1210
As first user plays the game, record and store data associated with the first user's actions to the server (e.g., data indicating actions taken, location, time stamp, etc.)
1215
Identify the number and location(s) for inserting ghosts associated with the first user on canvases displayed to other players
1220
Identify if the first user's ghost(s) is usable for each user
1225
For each player engaged in a gaming server, load all relevant information stored on the server
1230
Display ghost(s) on canvas for each user and configure the ghost to be "usable" by other players

Figure 12

Figure 13

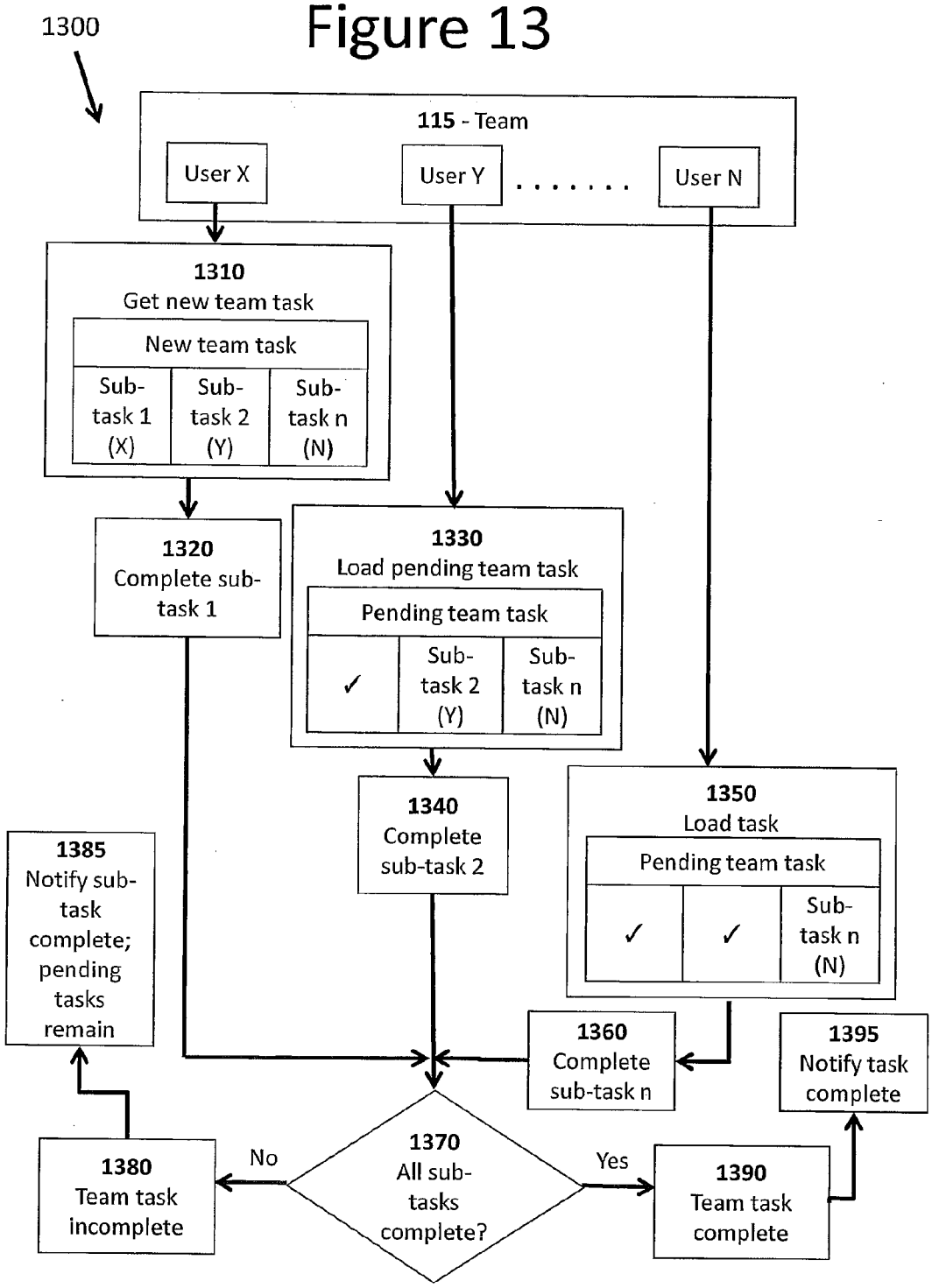
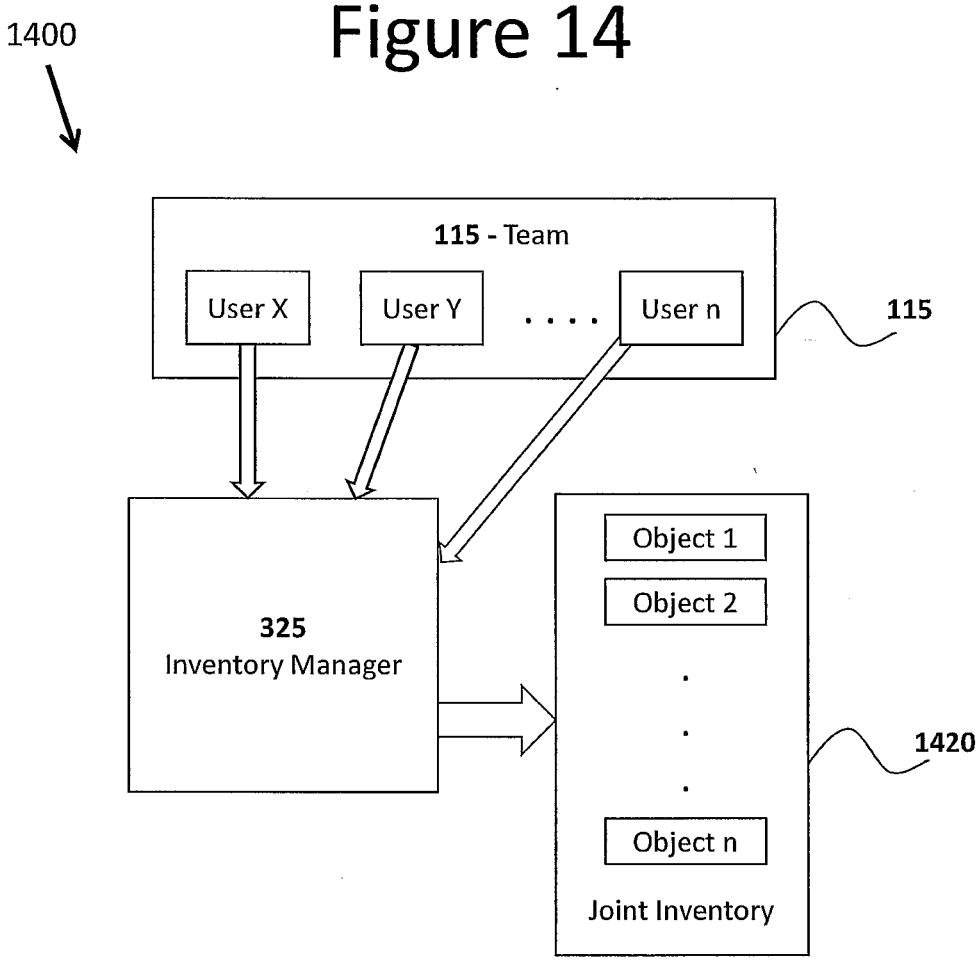

1300
115 - Team
User X
User Y
User N
1310
Get new team task
New team task
Sub-task 1 (X)
Sub-task 2 (Y)
Sub-task n (N)
1320
Complete sub-task 1
1330
Load pending team task
Pending team task
✓
Sub-task 2 (Y)
Sub-task n (N)
1340
Complete sub-task 2
1350
Load task
Pending team task
✓
✓
Sub-task n (N)
1385
Notify sub-task complete; pending tasks remain
1360
Complete sub-task n
1395
Notify task complete
1380
Team task incomplete
No
1370
All sub-tasks complete?
Yes
1390
Team task complete

Figure 14


1400
115 - Team
User X
User Y
. . . .
User n
115
325
Inventory Manager
Object 1
Object 2
Object n
Joint Inventory
1420

ASYNCHRONOUS TEAM-BASED GAME

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application is being filed concurrently with U.S. patent application Ser. No. [TBD] (Attorney Docket: 264-003) titled "USABLE GHOSTING FEATURES IN A TEAM-BASED GAME", the entirety of which is herein incorporated by reference.

COPYRIGHT NOTICE

[0002] A portion of the disclosure of this patent document contains material, which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

FIELD OF THE INVENTION

[0003] The present principles are directed to playing games over a network, and more particularly, to an asynchronous, multiplayer gaming platform.

BACKGROUND OF THE INVENTION

[0004] A variety of different game types are currently in use that permit users to play as a team. For example, games have been developed for personal gaming consoles (e.g., Microsoft's Xbox™, Sony's Playstation™ or Nintendo's Wii™), personal computers, handheld gaming devices and mobile devices (e.g., cell phones and personal digital assistants). Many of these games can be played over a network such as the Internet. The games can be classified into different categories.

[0005] One category of games is referred to as massive multiplayer online games (MMO or MMOG). These games can support hundreds, or even thousands, of users playing a game over the Internet, and these games are typically played from a personal computer. An example of such game is the well-known World of Warcraft™ developed by Blizzard Entertainment.

[0006] Personal gaming consoles, such as Xbox™ and Playstation 3™, provide functionalities which permit users to connect to a network and play games over the Internet. For example, Xbox Live™ is an online multiplayer gaming service offered by Microsoft™ to users at an additional cost.

[0007] Social networking games represent another category of games which are played over the Internet. These types of games are played through social networks. Exemplary social networking games include Farm Town™, Green Patch™, Mob Wars™ and The Sims Social™.

[0008] Although the idea of playing a game as a team may exist on current platforms, existing platforms only permit team play in a synchronous manner.

[0009] Furthermore, it is pointed that some of the games described above may implement "ghosting" features. Generally speaking, the ghosting features indicate to a user of a team the actions which were engaged in by other players on the user's team. However, while the ghosting features provided in existing team games may provide some indication as to the actions taken by other team members, existing team games do not permit users to interact or utilize the ghosting features during gameplay.

SUMMARY OF THE INVENTION

[0010] The present disclosure relates a new genre of gaming which facilitates a novel level of collaboration among players that form a virtual team. Amongst other things, members of the virtual team can cooperate to complete team tasks, exercise joint ownership over virtual goods in a shared inventory, and jointly earn team accolades. Users may play the game asynchronously at different times. Every time a user plays the game, data associated with the user's gaming session is recorded on a centralized server which is accessible to players on the user's team. This data may be utilized to generate and present ghosting features on a gaming canvas displayed to other team members. Amongst other things, the ghosting features may indicate gaming activities engaged in by the user, and can be utilized by the other team members in various ways (e.g., to complete tasks, obtain objects or earn accolades).

[0011] In accordance with the present principles, a method is disclosed for providing an asynchronous game. Shared elements (e.g., a shared canvas, shared object or a shared inventory) of a game are stored on a computer readable storage medium that is accessible to the players of the team over the network. Data is received from a first player of the team which is indicative of gaming activities modifying the shared elements. The shared elements are updated with the data received from the first player to generate an updated game state. Update data representing the updated game state is transmitted to at least one other player on the team.

[0012] In accordance with another embodiment of the present principles, a system disclosed for providing an asynchronous game. A memory stores shared elements of a game that is accessible to players of a team over a network. A server is configured to receive data from a first player of the team indicative of gaming activities altering or modifying the shared elements, and to update the shared elements with the data received from the first player to generate an updated game state. The server is further configured to transmit update data representing the updated game state to at least one other player on the team.

[0013] In accordance with an embodiment of the present princples, a method is disclosed for providing a ghost in an asynchronous game. Data indicative of gaming activities engaged in by a first user is recorded during asynchronous gameplay. At least one ghost to be displayed to a second user is generated with the recorded data associated with the first user. The at least one ghost is configured for use by the second user in performing at least one action.

[0014] In accordance with an embodiment of the present princples, a system is disclosed for providing a ghost in an asynchronous game. The system includes a database for storing recorded data indicative of gaming activities engaged in by a first user during asynchronous gameplay. A ghost controller is configured to utilize the recorded data associated with the first user to generate at least one ghost for display to a second user, and to configure the at least one ghost for use by the second user in performing at least one action.

[0015] These and other features and advantages will become apparent from the following detailed description of illustrative embodiments thereof, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF DRAWINGS

[0016] The invention is illustrated in the figures of the accompanying drawings which are meant to be exemplary

and not limiting, in which like references are intended to refer to like or corresponding parts, and in which:

[0017] FIG. **1** is a system for implementing an asynchronous, team-based gaming platform in accordance with one embodiment of the present principles.

[0018] FIG. **2** is a detailed view of a client for facilitating an asynchronous gaming platform in accordance with an embodiment of the present principles.

[0019] FIG. **3** is a detailed view of a server for facilitating an asynchronous gaming platform in accordance with an embodiment of the present principles.

[0020] FIG. **4** is a method for retrieving an updated game status at a client in accordance with an embodiment of the present principles.

[0021] FIG. **5** is a method for transmitting an updated game status to a server in accordance with an embodiment of the present principles.

[0022] FIG. **6** is a method for operating an asynchronous, team-based gaming platform in accordance with an embodiment of the present principles.

[0023] FIG. **7** is a method for creating a team in accordance with an embodiment of the present principles.

[0024] FIG. **8** is a method that illustrates how a shared environment can be implemented between two players in accordance with an embodiment of the present principles.

[0025] FIG. **9** is a canvas for a game that illustrates an exemplary manner of using a ghost of a teammate during gameplay.

[0026] FIG. **10** is a canvas for a game that includes ghosting features to indicate actions taken by teammates or other players in accordance with one embodiment of the present principles.

[0027] FIG. **11** is a canvas for a game that includes ghosting features to indicate actions taken by teammates or other players in accordance with another embodiment of the present principles.

[0028] FIG. **12** a method for inserting a ghost into a canvas in accordance with an embodiment of the present principles.

[0029] FIG. **13** a diagram illustrating a team task that is distributed among a plurality of team members in accordance with an embodiment of the present principles.

[0030] FIG. **14** a system for providing a joint inventory that is shared among a plurality of team members in accordance with an embodiment of the present principles.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0031] In the following description, reference is made to the accompanying drawings that form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the present invention.

[0032] Referring now to the drawings in which like numerals represent the same or similar elements and initially to FIG. **1**, a system **100** is disclosed for implementing an asynchronous, team-based gaming platform in accordance with an embodiment of the present principles. As illustrated in this figure, the system **100** includes a server **120** that is connected to a plurality of clients **110** via a network **140** (e.g., the Internet, a local area network (LAN), intranet or other network). Clients **110** connect to the server **120** via the network **140** to play the game **125**. Clients **110** may connect to the server **120** using any type of wired or wireless connection.

[0033] In an embodiment, the game **125** may be embodied as a program, application or module associated that includes a set of instructions that can be executed by a processor. The game **125** may be any type of game, e.g., a military game, strategy game, puzzle, city building game, simulation game, etc. The game **125** can be played in variety of different formats including: (1) an "ad-hoc format" in which users join teams to accomplish a specific task or mission; (2) an "exclusive team format" in which each user is a member of a single team at any time; (3) a "non-exclusive format" in which a user can be part of multiple teams at a given time; (4) a "one time format" in which users that are already playing the game can join as a team for a particular session; or (5) a "team vs. team" format in which a first team of players compete or collaborate with one or more additional teams. Other formats and game types may also be utilized.

[0034] A plurality of clients **110** may collaboratively play the game **125** together as a virtual team **115**. Users that are part of a virtual team **115** may work together or cooperate to complete tasks, missions or achievements. In addition, virtual teams **115** may share an inventory (e.g., weapons or health elixirs), and exercise joint ownership over virtual goods (e.g., can buy, sell, upgrade or use items in the inventory). While each user can still have individual tasks or achievements, users are part of a larger team-based gameplay progression.

[0035] A team **115** can move between different games **115**. Hence, the notion of a team **115** is not necessarily based on a specific game. Rather, a team **115** may play several games **125** together. In certain embodiments, the information associated with a team **115** in one game **125** can be accessed and utilized by other games **125**.

[0036] No direct connection between the team members is required to create a virtual team **115**. However, in certain embodiments, the clients **110** may be directly connected each other (e.g., using a wired or wireless connection). In addition, when playing the game as part of a virtual team **115**, it is not necessary for all members of the team to be playing the game **125** simultaneously. Although users are permitted to play the game **125** at the same time, asynchronous gameplay is provided to permit users to play the game at different times. Each user utilizes a client **110** to connect to the server **120** to play the game **125**. When starting the game **125**, the client **110** retrieves the current status of the game **125**. Amongst other things, the data retrieved includes all of the relevant actions, activities, tasks, achieved progress, outcomes, etc. undertaken by the user's teammates while the user was not connected to the game **125**.

[0037] As the user plays the game **125**, all data associated with the user's gaming session may be recorded by the user's client **110** and stored on the server **120**. When other users (e.g., teammates) play the game **125**, the recorded data may be retrieved and utilized to generate and display ghosting features on the gaming canvas displayed to other team members. Amongst other things, the ghosting features may indicate gaming activities engaged in by the user. The ghosting features be utilized by the other team members in various ways (e.g., to complete tasks, obtain objects or earn accolades).

[0038] The data recorded and stored for a user's session may include data associated with a variety of different gaming activities. As used herein, a "gaming activity" generally refers to any type of action or activity the user engages in

within the confines of the game **125**. For example, a gaming activity may represent the situation where a user progresses to a new level in a game **125**, engages in battle with an opponent, unlocks an achievement, completes a task or sub-task assigned to the user, utilizes a virtual good or item in the inventory, or any other activity in which the user engages in while playing the game **125**.

[0039] As a user plays the game **125**, everything that the user does may be recorded, including all gaming activities engaged in the user, and the information associated with each recorded session may be stored on the server database **130** (although the client **110** may temporarily save this data locally before transmission to the server **120**). When saving information related to a user's session to the database **130**, a variety of different parameters may be saved with each activity including time stamps, information indicating the type of activity that took place (e.g., opened a crate, opened a door or added a building), specific parameters associated with the activity (e.g., an identification number for a door that was opened), movement data (e.g., vectors and locations), information indicating the speed of an activity or player, outcomes related to the data (e.g., whether a trophy was achieved or whether a new level was attained), current location of the player or activity, game specific data (e.g., pertaining to a task or level), or objects used in performing the activity (e.g., a club or car). Any other data related to a gaming activity may also be saved to the database **130**. In certain embodiments, this data may be saved based on time intervals (e.g., every few seconds all activity data is uploaded to the server **120**) or based on the occurrence of particular actions or activities (e.g., activity data is uploaded to the server **120** after the user completes particular activities or tasks).

[0040] At any moment, the server **120** stores all of the users' gaming activities (except for possibly the last few actions that are yet to be transmitted). This information is referred to as the "current state" or "current status" of a game. The current status of a game may be retrieved by a client **110** when starting a game **125** and periodically as a user plays the game **125**.

[0041] FIGS. **2** and **3** illustrate a detailed implementation of a server **120** and a client **110** in accordance with an embodiment of the present invention. While the server **120** and the client **110** may generally be described in terms of modules which represent separate and distinct entities, it should be recognized that the functions performed by the modules may overlap and that the modules may be combined in in any manner whatsoever (e.g., all of the modules may be implemented as part of a single program). While the modules may be described as software modules, it should be recognized that each of the modules may be implemented in software, hardware or any combination of the two. Furthermore, in certain embodiments, components or modules running on the server **120** may alternatively be performed by the client **110**, and vice versa.

[0042] Initially addressing FIG. **2**, an exemplary system **200** for implementing a client **110** is disclosed. The client **110** may include a content retriever **205**, a content updater **210**, a local memory **215** and a merger **220**. In certain embodiments, the client **110** and its related components may be implemented as a piece of software that is run on a device (e.g., desktop computer, tablet, cell phone, personal digital assistant, laptop, etc.) which is operated by a user. The client **110** may represent a website, flash module, game disk, (e.g., a compact disk or computer disk), mobile application, console game, or the like. All user interaction with the game **125** is done through the client **110**. Some actions are handled locally by the client **110**, while other are sent to the server **120** that processes the associated data and returns a response.

[0043] The content retriever **205** may represent an application, program routine or software module that is configured to pull data from the server **120** to obtain an updated status of a game **125**, or may be configured to receive data that is pushed to the client **110** indicating the updated status of the game **125**. The content retriever **205** may be responsible for all communication between the client **110** and the server **120**, as well as any communication with external entities (e.g., the website for Facebook™ or other social network). Data retrieved from the server **120** may be stored in a local memory **215**, which may represent a data structure or collection of data that is stored on a storage medium (e.g., RAM) residing on the same device that is running the client **110**.

[0044] A merger **220** may represent an application, program routine or software module that is configured to analyze the retrieved content to determine the content that is to be displayed by the client **110** to the user. Specifically, the merger **220** merges all of the various data associated with the game **125** into a single presentation for display. For example, the merger **220** consolidates all the data that may affect the appearance of the canvas and the use of a team's inventory. This may include consolidating all data associated with providing ghosting features (discussed below) or other elements on the canvas.

[0045] The merger **220** is also configured to correctly determine and present information that is relevant to a particular user. Certain information may be relevant to one team member, while it is not relevant for other team members. The merger **220** identifies and incorporates the information which is relevant for a particular user into a unified presentation. The unified presentation produced by the merger **220** is then provided to be displayed.

[0046] As a user plays the game **125**, data associated with the users' activities may be recorded in the local memory **215**. The content updater **210** may retrieve this data from the local memory **215** and transmit the data to the server **120** as the game **125** is being played or when a user terminates a gaming session. The data may be transmitted periodically or upon the occurrence of particular activities or actions that occur during the game **125**.

[0047] FIG. **4** illustrates an exemplary method **400** for configuring the content retriever **205**. The content retriever **205** may represent an application, program routine or software module that is configured to retrieve data from the server **120** indicating the current status of the game (block **401**). As the user plays the game **125**, the client **110** sends a request to a server **120** for updates associated with the game **125** (block **405**). The request may be sent periodically (e.g., every few seconds), in response to certain gaming activities (e.g., when a new level is achieved), in response to an explicit indication by the user (e.g., the user selects a "retrieve updates" button) or for other reasons.

[0048] A determination is made as to whether updates are available (block **410**). If no updates are available, then the method **400** proceeds back to block **405**. However, if updates are available, then the updated data is retrieved (block **415**). Any retrieved updates may be stored in a local memory **215** by the content retriever **205**. A merger **220** then analyzes the retrieved data and decides what relevant status needs to be

presented for displaying the current status of the game **125** (block **420**). The method **400** proceeds back to block **405**.

[0049] FIG. **5** illustrates an exemplary method **500** of operation for the content updater **210**. A user loads the game **125** via the client **110** to begin gameplay (block **505**). As the user plays the game **125**, all gaming activities associated with the game **125** are recorded by the client **110** (block **510**). Data associated with the activities may be stored a local memory **215** by the client **110**.

[0050] A determination is made as to whether there are any updates to be transmitted to the server **120** (block **515**). It should be noted that it may not be necessary to transmit all data which is indicative of a user's activities. Rather, in certain embodiments, only certain activities require updates to be sent to the server **120**. If there are no updates to be transmitted, the method **500** proceeds back to block **510**. On the other hand, if updates are available, then the updates are transmitted to the server **120** for storage (block **52**) and the method proceeds back to block **510**.

[0051] Having described the operation of the client **110** and its associated components, an exemplary system **300** for implementing the server **120** is now provided with reference to FIG. **3**. As shown therein, the server **120** may include a memory **301** that includes a plurality of software components, applications or modules which are executable by a processor **360**. An asynchronous team play controller **305** is configured to manage and facilitate the operations relating to providing an asynchronous gaming platform for users that play a game **125** as part of virtual team **115**. For example, the asynchronous team play controller **305** may be responsible for storing updates received from clients in the gaming database **350** at the server **120**, and for ensuring that each client **110** that is connected to the server **120** during gameplay maintains the current status of the game.

[0052] The asynchronous team play controller **305** may also be configured to handle gameplay when two or more team members are simultaneously logged in to the game **125**. While two or more team members are permitted to play the game **125** at the same time, the game **125** is played in an asynchronous manner. Depending on the game, the asynchronous team play controller **305** can transmit updates periodically or in response to certain criteria (e.g., requests from clients **110** or upon completion of certain gaming activities). When multiple clients **110** are playing the game simultaneously, it is likely the update data that should be transmitted to each of the clients **110** will differ. For example, if players A, B and C are playing the game, player A may only need the update data associated with players B and C, while player B may only need the update data associated with players A and C. The asynchronous team play controller **305** is configured to determine the appropriate updates to be sent to each client **110**, and to transmit the updates to each of the clients **110**.

[0053] One can imagine that problems may arise when multiple users are playing the game **125** simultaneously due to the fact that the gaming database **350** is continuously being updated by more than one user. Thus, to ensure that updates are made atomically and in the appropriate manner, the asynchronous team play controller **305** may utilize a locking mechanism (e.g., semaphores and/or mutex locks), which permits only one user to update the database at any given time (e.g., a time interval defined by a few milliseconds). The asynchronous team play controller **305** resolves any potential activity "collisions" if two or more users are trying to send colliding data to the server **120**.

[0054] The team controller **310** is configured to handle operations related to managing a virtual team **115**. Depending upon the particular game **125** being played and make-up of a particular team **115**, the configuration of the team controller **310** may vary. The team controller **310** may be configured to add members to a team **115**, remove members from a team **115**, or edit existing members that are part of a team **115**.

[0055] Before a user is permitted to play a game **125**, the user may be required to setup a user profile **353**. In other cases, the server **120** automatically creates a user profile **353** for a user. For example, the server **120** may automatically generate a user entity and associated user profile **353** when a user enters a game **125**, or the server **120** may utilize information from an external entity (e.g., Facebook™) to establish a user profile **353**. The user profile **353** includes descriptive information about a user (e.g., name, location, user ID, email address, etc.) and may also specify gaming content (e.g., avatars) that is to be associated with the user during gameplay.

[0056] When adding or removing users from a team **115**, the team controller **310** may associate or disassociate a user profile **353** with an instance of a team profile **354** stored in the gaming database **350**. The team profile **354** may store all data related to a team (e.g., team members, games being played by the team, etc.) Likewise, when editing an existing member of a team **115**, the team controller **310** may modify the team profile **354** and/or user's profile **353** stored in the gaming database **350**.

[0057] The team controller **353** may also be responsible for enforcing a set of rules associated with a particular team or game. For example, consider an exemplary game **125** that requires three team members in order to play and which assigns each of the players to a specific role (e.g., a wizard, warrior and princess) that has abilities and functions which are unique to the role. In this scenario, the team controller **353** may be responsible for enforcing rules on the composition of the team (e.g., there must be at least three team members) and the rules associated with the roles assigned to each team member (e.g., the wizard can attack with magic and the warrior can attack with a sword).

[0058] In certain embodiments, the team controller **310** may also be responsible for the manner in which a team interacts with mercenaries. Generally speaking, the term "mercenary" refers to a user that is not a part of the team **115** or social graph, but who plays with the team for specific reason. A mercenary may be hired for a specific period of time, for a specific task or to assist with specific gaming activities. In certain embodiments, mercenaries may be hired and paid with virtual goods to fulfill a temporary or permanent position. For example, User X and User Y may play a game **125** as a team **115**. To complete a particular level, User Z may be added to the team **115** as a mercenary to assist User X and User Y. To reward User Z for participating and assisting the team **115**, User X and User Y may pay User Z with virtual goods in the team's joint inventory. After completing the level, User Z is no longer part of the team **115**.

[0059] A mercenary is typically controlled by a human user. However, in certain embodiments, a mercenary may not be controlled by a human user, but rather controlled by the game **125**, a program or artificial intelligence.

[0060] Virtual teams may play the game **125** on a "shared canvas" **352** which is controlled, maintained and managed by a shared canvas controller **315**. Generally speaking, the canvas **352** represents a drawing board and associated set of rules that permit a team of players to jointly carry out tasks or

jointly manage (e.g., own, edit, manage, delete, upgrade, use, grow, etc.) elements on the canvas. The canvas **352** is the playing ground for a game, and is similar to the well-known Tamagotchi™ digital pet in the sense that it may progress, change, produce outputs, and require a user to take care of it. The canvas **352** can be a 3D environment, a 2.5D isometric ground, a flat 2D environment, a drawing board, a game board, or other similar gaming environment.

[0061] The canvas **352** is typically tailored to the particular type of game **125** being played. For example, in the scenario where the game **125** being played is a game **125** directed toward constructing cities, the canvas **352** may represent a piece of land where users insert buildings and roads. Alternatively, in the case that the game **125** being played is a military game, the canvas **352** may represent a battlefield where players engage in battle.

[0062] In certain embodiments, when players form a virtual team **115**, all of the players comprising the team play the game on the same shared canvas **352** (assuming that the particular game being played includes a shared canvas **352**). The shared canvas **352** allows all the team members to interact with the canvas as if they were playing a single player game. However, instead of having one user that plays the game on a single canvas, a team plays the game **125** on a single joint canvas **352**. When a team member engages in gaming activities or actions that alter or modify the canvas **352**, the shared canvas controller **315** ensures that the changes are displayed to everyone on the team.

[0063] Depending upon the particular game, teams may be permitted to play the game **125** together on the same canvas **352** when the game **125** is being played in "team vs. team" mode. Regardless of how many teams **115** are sharing a single canvas **352**, the shared canvas controller **315** consistently maintains the canvas **352** for each player and ensures that all changes to the canvas **352** are displayed to everyone playing the game **125**.

[0064] In addition, if two or more members of a team are playing the game **125** at the same time, the shared canvas controller **315** may assist in periodically providing updates to the canvas **352**. Updates to the canvas **352** can be pushed to clients **110** or can be transmitted to clients **110** in response to received requests. The role of the shared canvas controller **315** may overlap with the role of the asynchronous team play controller **305** to some extent. While the update information is being transmitted to the server **120** for a specific user, the shared canvas controller **315** may lock the access to the canvas **352** to prevent other users from updating the canvas **352**.

[0065] For example, consider the situation in which two users are both playing the game **125** at the same time. If the first user performs an action that affects the canvas **352**, the shared canvas controller **315** identifies the action as one that requires updating the canvas **352**. Thereafter, the shared canvas controller **315** may lock access to the canvas **352** while the update is performed (thus, preventing the second user from updating the canvas at the same time) and unlock access to the canvas **352** after the canvas is updated. The shared canvas controller **315** may also ensure that the canvas **352** displayed to the second user is updated accordingly by transmitting the update information to the second user.

[0066] To further illustrate the concept of a shared canvas **352**, consider the exemplary shared canvas **352** depicted in FIG. **11** (which is discussed in further detailed below in describing ghosting features of the game **125**). The canvas **352** is provided for a military game and is being played by a team of three members consisting of Players A, B and C. Player A controls an army **1110**, Player B controls an air force **1120**, and Player C controls a navy **1130**. The members of the team **115** work together to conquer and control a group of cities **1140** that are defended by enemy forces.

[0067] Assume the canvas **352** is being displayed to Player A, who is in control of the team's army **1110**. When Player A performs an activity which alters the canvas (e.g., moves the army to a new location or attacks opposing forces), the client **110** being utilized by Player A records all of the activities and transmits data associated with the activities to the server **120** for storage in the database **350**. When Player B or Player C loads the game **125** at a later time, the current status of the game **125** is retrieved from the server **120** and Players B and C are be able to see all changes which have been made to the canvas **110** by player A. If Players B and C are playing the game **125** simultaneously with Player A when the server **120** updates the game data **351** or shared canvas **352** with the data received from Player A, the shared canvas controller **325** ensures that their associated clients **110** maintain the current status of the game **125** during gameplay by transmitting updates to Players B and C.

[0068] The shared canvas controller **315** may work in conjunction with the ghost controller **320** to determine operations relating to "ghosting". As used herein, a "ghost" represents the aggregation and playback of the actions a user or teammate has performed while playing a game **125**. Ghosts can be used for a wide variety of different purposes. For example, ghosts can be used to indicate a recent location or action undertaken by another player, to assist a user in completing a task, to assist a user in reaching new areas of the board that are not accessible without team collaboration, to assist a player in earning accolades, or to assist a player in acquiring an object. As used herein, a "usable ghost" refers to any ghost that can be interacted with by a user playing the game **125**, or that can be utilized by the player to perform certain actions, activities or tasks.

[0069] Staying with the above example, when Player A loads the game **125** and is provided with the exemplary canvas **352** illustrated in FIG. **11**, Player A can view a ghost **1120** for Player B and a ghost **1130** for Player C. In this particular game **125**, multiple ghosts are included for each teammate of Player A. The ghosts **1120** and **1130** are included at the most recent location of the players (e.g., the last location before ending a gaming session or, if the player is playing simultaneously with Player A, the last location of the player which was transmitted to the server **120**). Additional ghosts may also be included on a canvas **352** that is displayed to Player A at a location where gaming activities that were previously undertaken by Players B and C occured. For example, a ghost **1121** is included on the canvas **352** at location in the North Atlantic indicating that Player B previously deployed planes to attack the enemy's navy. Similarly, a ghost **1131** of Player C is included at location south of Australia indicating that a naval battle took place between Player C and the enemy.

[0070] Any of the ghosts displayed to Player A may permit Player A to view the activities previously engaged in by Player B or Player C. For example, if Player A moves his army during gameplay near the ghost **1121** of Player B, Player A is able to view the Player B's air force attacking the enemy's navy, thus giving Player A the feeling that he is playing together in real-time or in real collaboration with Player B.

Likewise, the ghost **1131** of Player C may replay the actions that were undertaken by Player C while engaged in the naval battle.

[0071] Any of the ghosts inserted into the canvas **352** of FIG. **11** can be configured to be usable by other players. For example, the ghosts may be utilized to assist a user in attacking enemy forces or to transport a user's forces to a new area on the canvas.

[0072] Further details regarding the operation of the ghost controller **320** are discussed below with reference to FIGS. **9** through **12**.

[0073] The inventory manager **325** is configured to maintain a joint inventory of virtual goods for members of a team **115**. Virtual goods are non-physical objects that may be acquired while playing the game **125**. Depending upon the particular game, the virtual goods in a team's inventory may vary. For example, the joint inventory may include ammunition or weapons in a military game, while the joint inventory for a city constructing game may includes bricks, concrete and other materials to be used in constructing a city.

[0074] Virtual goods can be acquired in various ways including, but not limited to, using virtual currency, using real money in a game store, receiving a gift from friend, requesting a gift and receiving a positive reply, completing a task or mission, picking up or activating objects which randomly appear when playing the game, jointly purchasing a virtual good, obtaining an achievement or trophy in game, virtually betting on an item with another team or user, or by responding to a wall post in a social forum. Virtual goods can be obtained in other ways as well.

[0075] The inventory manager **325** facilitates team ownership of virtual goods by permitting players to share all or part of their inventory. When the inventory is shared, virtual goods acquired by a team member are available for all other members of team **115**. Each member of the team **115** will be able to use, place, delete, sell, upgrade, etc. a shared virtual good as if the was his own. However, in certain embodiments, there may be some limitations placed in each game **125** to avoid loss of virtual goods.

[0076] In some embodiments, virtual goods in the inventory may include "living virtual goods". A user may be permitted to use, develop or otherwise interact with the living virtual goods. An example of a living shared virtual good would be a dog, acquired by User X and placed in his joint apartment with User Y. Both User X and Y can now feed the dog, watch the dog grow, play with the dog and share the responsibility in jointly owning a dog.

[0077] In some embodiments, the inventory may also include "jointly acquired virtual goods". Jointly acquired virtual goods represent virtual goods that can only be obtained in a joint effort by all (or part) of the team members. Jointly acquired virtual goods can be obtained in various ways. For example, a virtual good can be jointly purchased by team members using real money or virtual money from a game store associated with a game **125** (e.g., each team member buys a piece of a virtual good), or can be obtained as gifts or prizes from jointly completing tasks.

[0078] To further illustrate the concept of a jointly acquired virtual good, consider a team **115** that is seeking to purchase a car together. User X buys the car body, but is not permitted to purchase the tires or the engine. User Y obtains the tires as a prize for completing a task, and subsequently acquires the engine from the game store. Each time a portion of the jointly acquired virtual good is acquired, a notification may be transmitted (e.g., by the socializer **340**) to the players on the team. After the team has obtained all the necessary pieces of the car, the car is completed and can be used by any of the team members.

[0079] The task manager **330** is responsible for managing tasks or missions that are part of the game **115**. The task manager **330** is configured to handle both individual tasks and team tasks. An individual task represents a task that can be completed by a single player. A single task may be comprised of a set of sub-tasks. For example, a single task may comprise the following sub-tasks: gather five logs; get a pipe from you friend; send 2 gifts to friends.

[0080] A team task represents a task that is completed by the actions of two or more players. A team task may be divided into sub-tasks that are completed by different team members. The team tasks can be can be comprised of a regular group of sub-tasks that can be achieved together. For example, any player can contribute to any of the sub-tasks relating to gathering five logs, retrieving a pipe and sending a gift.

[0081] In other embodiments, the sub-tasks that comprise a team task may only be able to be performed by certain team members based on the role assigned to a team member. For example, consider a game that includes a task that involves breaking into a locked room. Player A is assigned the role of a locksmith, while player B is assigned the role of a trained soldier. In order to break into the room and complete the task, the team has to complete sub-tasks such as disarming a guard and picking a lock to the room. The task manager **330** may configure the task such that Player B has to neutralize the guard, while Player A has to pick the lock. When both players complete the sub-tasks assigned to their character, the team task is complete.

[0082] In certain embodiments, a team task (like many other elements in the game **125**) is asynchronous, and can be completed by different users in no particular order or in no particular period of time. In other embodiments, the sub-tasks associated with a particular team task have to be executed in a particular order to complete the team task. Trophies, achievements, bonuses, etc. may be unlocked or awarded to players upon the completion of a task. If the task is a team task, the same can be awarded to the team.

[0083] The content management system **335** represents a set of administrative functions that permits a game creator, programmer or other person to edit, insert or remove certain features that are part of a game **125**. For example, the content management system **335** may include functions for editing the canvas, virtual goods, tasks, advertisements, tutorials or help features in a particular game **125**. The content management system **335** may perform additional operations related to the content of a game **125**.

[0084] The socializer **340** is configured to handle operations relating to facilitating communication between a player and at least one other person. The socializer **340** may permit certain types of communication among teammates, opposing teams, or even persons have not played the game. The socializer may be configured to transmit notifications or communications automatically in response to a user activity during gamgeplay, or may wait for a specific on-demand socializing request from a user (e.g., when a user clicks on a "share my score" button).

[0085] Amongst other functions, the socializer **340** may be configured to: (1) transmit invitations (e.g., a player can invite other players or non-players to start a game, complete a tasks

or engage in some type of gaming activity); (2) post information on the wall of a user in a social forum (e.g., "click here to play" or "send virtual good"); (3) send gifts or virtual goods to a friend or group of friends; (4) send notifications to players or teams (e.g., indicating a new call for action or indicating that information was posted on the user's wall); or (5) provide a ticker function which provides a short textual notification about gaming activities. The socializer **340** may be configured to provide additional means of communicating as well.

[0086] In certain embodiments, the server **120** may further include a communication adaptor **345**. The communication adaptor **345** permits the back-end or server **120** to communicate with clients **110** of different types (e.g., clients for a desktop computer, tablet, cell phone, personal digital assistant, laptop, etc.). For example, the communication adaptor **345** may utilize an adaption layer to translate and adapt communications between the server **120** and the client **110** according to the particular type of client **110** be utilized by a user. In this manner, the communication adaptor **345** assists in providing a "platform agnostic system" that is capable of communicating with various types of clients **110** from a single back-end system. FIG. **6** illustrates a general method **600** for playing a team-based asynchronous game in accordance with an embodiment of the present principles. The method **600** begins when a user initializes or creates a new game **125** (block **605**). A user may accomplish this by submitting a request via the user's client **110** to the server **120**.

[0087] After a game **125** has been initialized, a team **115** comprising at least two players is created (block **610**). The operations related block **610** may be implemented by the team controller **310** described above. During the creation of a team **115**, roles may be assigned to specific players. A role may include a set of parameters, functions, features, abilities, etc. which are not available to other players participating in the game **115**. Referring back to the example provided above with respect to FIG. **11**, three players may be assigned different roles (i.e., an army commander, an air force commander and a naval commander), and each of the roles may be associated with a distinct subset of functionalities and attributes (e.g., only the air force commander can attack with planes, and only the naval commander can move in water).

[0088] The creation of a team **115** may also include creating or selecting mercenaries to be a member of the team **115**. As explained above, a mercenary can either be an AI or another player which is not on the same team. In certain games, a mercenary may automatically be included as a member of a team regardless of the team's composition. In other games, a mercenary may be added to a team **115**.

[0089] FIG. **7** discloses one exemplary method **700** for configuring a team controller **310** to create a team **115**. One or more users can access a set of operations for establishing a team **115** (block **705**). Upon accessing the team setup operations, the user is presented with several options for a modifying a team (block **710**). Specifically, the user can choose to hire a mercenary team member (block **715**), to add a user to the team (block **725**) or to modify an existing team member (block **720**).

[0090] If the user opts to add a mercenary to the team (block **715**), the socializer **340** may transmit a notification to each of the team members. After hiring the mercenary, a new team entity is created for the team (block **745**), and the user is provided with a prompt or interface that inquires whether the user would like to continue modifying the team (block **750**).

[0091] If the user wishes to further modify the team **115**, the method proceeds back to block **710** and the user is presented with the options for further modifying the team **115**. If the user selects the option relating to adding a team member (block **725**), an invitation or notification is sent to the invitee (e.g., via electronic mail). If the invitee confirms the invitation, a new team entity is created for the invitee and the invitee is added to the team (block **745**). Each time a new entity is added to a team **115**, additional notifications may be transmitted to notify each of the team members.

[0092] A user can also choose to modify an existing member (block **720**). Specifically, the user may be presented with options for removing a member from a team **115** (block **730**) or for editing data (e.g., avatar, role, name, etc.) associated with an existing team member (block **735**).

[0093] At some point, the user determines that no further modifications are necessary (block **750**) and the team is ready for gameplay (block **755**)

[0094] Referring back to the method **600** illustrated in FIG. **6**, after the team **115** is ready to begin gameplay, an asynchronous multiplayer game **125** is created (block **615**). The operations associated with creating the game **125** may be carried out by the asynchronous team player controller **305** described above in cooperation with other modules.

[0095] When creating the game **125**, the server **120** may initialize a shared canvas **352** that provides a platform or virtual game board for users to play the game **125**. This may include inserting avatars into the canvas **352** for each player that is a member of the team and determining a set of rules to control the interaction between the players. The creation of the game **125** may further include initializing a joint inventory of virtual goods which can be jointly used, managed, manipulated, etc. by each of the players on the team **115**, establishing team tasks which can be performed by the players when playing the game **125**, and configuring the canvas to display ghosting features. In performing these operations, the asynchronous team play controller **305** may cooperate or communicate with the team controller **310**, the shared canvas controller **315**, the ghost controller **320**, the task manager **330** or other entity described above.

[0096] Once the game **125** and associated parameters have been initialized, each member of the team can begin playing the game **125** (block **620**). Players are not required to play the game **125** at the same time. Rather, players can login into the game **125** at different times and complete tasks or sub-tasks as they please. In some cases, users are permitted to start the game without a team, and subsequently join or create a team at a later time.

[0097] As the user engages in gaming activities while playing the game **125**, the client **110** for each user records all associated information and transmits this information for storage on a server **120**. The gaming activity information being stored on the server **120** may be used to update the canvas **352**, joint inventory, tasks or other game data **351** associated with the game **125**.

[0098] FIG. **8** discloses an exemplary method **800** illustrating how two users, User X and User Y, can utilize the information stored on the server **120** to provide a shared canvas **352**, joint inventory, and team tasks.

[0099] The method **800** begins when User X opens the game **125** (block **805**). Upon opening the game **125**, the current status of the game is retrieved from the server **120** (block **810**). This may include downloading information to User X's client **110** that indicates the most up-to-date display

of a shared canvas **352**, the status of the team's joint inventory, and any information relating to the completion, or lack thereof, of tasks or sub-tasks.

[0100] Next, User X makes changes to the canvas **352** and objects in the team's inventory while playing the game (block **820**). For example, in a game **125** related to constructing cities, User X may add buildings to the shared canvas **352**, and may add construction materials (e.g., brick and steel) to the team's inventory. Data indicative of such gaming activities (as well as other gaming activities) is transmitted to the server **120** for storage (block **825**).

[0101] User Y then opens the game **125** (block **830**), and loads the current status of the game **135** including data relevant to the current state of the shared canvas **352**, shared inventory and team tasks (block **835**). After loading the game **125**, User Y engages in gaming activities that modify the shared canvas **352** (block **840**). For example, User Y may add a bridge to the canvas **352** that connects two pieces of land separated by a body of water. User Y also completes a portion of a team task. These activities are transmitted to the server **120** for storage (block **845**).

[0102] The current status of the game is stored on the server **120** at any given time (block **850**). At this point, the server **120** stores, inter alia, data that indicates the changes that have been made to the canvas **352** by Users X and Y, the updated inventory that was supplemented by User X, and the team task that was completed by User Y.

[0103] Further details are now provided with reference to FIGS. **9** thorough **12** to illustrate how ghosting features can be utilized in providing an asynchronous multiplayer game **125**. Any details discussed below with respect to providing the ghost features may be implemented by the ghost controller **320**.

[0104] As explained above, a ghost may be created from the recorded contents of a user's gaming session. A ghost that has been inserted into the canvas **352** for a particular user can be configured to interact with users and to assist users in performing a variety of different actions, activities or tasks. For example, a ghost may be inserted to assist a user in completing a task, to assist a user in acquiring a virtual good or object, to assist a user in completing a level or to assist a user in earning an accolade.

[0105] The representation of a ghost can vary dramatically. Ghosts can be represented statically (e.g., using a graphic) or dynamically (e.g., using an animation or video). For example, a static graphic may be inserted at a particular location to indicate to a user that the user's teammate completed part of task. In other examples, the ghost of a teammate may be dynamic in the sense that a ghost actually recreates the actions of a user, or in the sense that the ghost is configured to follow the user and trail the user at all times during gameplay. In either case, the user may interact with the ghost to complete the remaining portion of the task.

[0106] FIG. **9**, which illustrates an exemplary canvas **352** for a game **125**, demonstrates how a ghost of a teammate can be utilized by a user in a game **125**. In the game depicted therein, assume that User X is playing the game **125** on a team **115** with User Y. In this level of the game **125**, the team **115** is running through the canvas **352** and breaking any boxes **930** which they come across in order to maximize the team's score. However, in order to break a box **930**, a box **930** must be kicked by at least two team members.

[0107] During a previous gaming session, User Y ran through the board and kicked each box once. All of User Y's activities were recorded while User Y was playing the game and were stored on a server **120**. Subsequently, when User X begins playing the level illustrated in FIG. **9**, the information stored by User Y is retrieved and utilized to recreate the actions undertaken by User Y. Specifically, ghost controller **320** displays a ghost **920** of User Y and configures the ghost to re-run the board in the same or similar manner as previously done while User Y was playing the game. User X controls his character or avatar **910** to follow the ghost **920** of User Y. Each time User X sees the ghost **920** of User Y kick one of the boxes **930**, User X may instruct the avatar **910** to do the same. When a box **930** is kicked a second time by User X, the box **930** will break and the team **115** will be awarded points.

[0108] In certain embodiments, **320** the ghost controller may include a time shifter application or module that is configured to shift ghost actions forward or backwards in time to create a better synchronization and gameplay and to allow the user to avoid "losing" the ghost when playing the game **125**. For example, in the game **125** illustrated in FIG. **9**, the ghost **920** of User Y may be shifted back in time if it the ghost **920** runs too far ahead of User X during gameplay. In other cases, such as when User X is running faster that the ghost **920** of User Y, the ghost **920** of User Y may be shifted forward in time.

[0109] Better gameplay may also be provided by synching the activities of the different ghosts and the user. For example, if a box was broken by a first ghost, there is no need to show it being destroyed again by another ghost. Therefore, the ghost controller **320** may ensure that all activities are synched and makes sense in terms of game play.

[0110] The above example demonstrates how the ghost **920** of a teammate can be configured to assist a user in completing a task or earning an accolade. It should be recognized that this one simple example and that the manner in which ghosts are configured to be usable can vary greatly from game to game.

[0111] Consider the example discussed above with reference to FIG. **11** once again. Player A may utilize any of the ghosts for Player B or Player C during gameplay. In this example, Player A may move the army under his control to the location of one of the ghosts **1120** and **1130** to use one of the ghosts to transport Player A's army across the Atlantic Ocean. For example, Player A may move his army to Location X. Once at Location X, the army may board the transport ships in Player C's navy and utilize the ghost **1130** of Player C to transport the army across the Atlantic Ocean.

[0112] In certain embodiments, the ghosting controller **320** may also insert ghosts on a canvas **352** in a location where a user has engaged in a particular gaming activity (e.g., unlocked a door, complete a task or sub-task, or received a trophy). In determining whether to insert a ghost to indicate activities engaged in by a player, the ghost controller **320** may analyze a list of predetermined triggers that indicate if a ghost is to be inserted into the canvas **352** for a particular activity.

[0113] FIG. **11** includes two additional ghosts **1121** and **1131** that indicate activities recently engaged in by Players B and C. A ghost **1121** located in the northern part of the Atlantic Ocean indicates that Player B attacked a fleet of the enemy's naval ships with aircraft under the control of Player B. Similarly, a ghost **1131** located south of Australia indicates that Player C engaged in a naval battle with the enemy. Using the data stored on the server **120** by the teammates, the user may be permitted to view the previous actions of teammates as if they were occurring in real-time. The user may further be permitted to interact with the ghosts to perform gaming

activities such as completing tasks or obtaining objects (or any other gaming activity). In certain embodiments, a user may be permitted to view additional details relating to an activity engaged in by a teammate by scrolling over or clicking on the ghosts.

[0114] The ghost controller **320** may utilize certain criteria or gaming activities to determine the manner in which a ghost can be utilized by a user. For example, the ghost controller **320** may determine that the ghost **1130** of Player C is usable by Player A when the ghost **1130** is located within a predetermined distance from the land. Since Player C's ghost **1130** is located directly adjacent to the tip of Africa on the canvas **352**, Player A may move his army to Location X and utilize the ghost of Player C to transport his army across the Atlantic Ocean. However, if Player C's ghost **1130** was not located near a piece of land which is accessible by Player A, the ghost controller **320** may determine that the ghost **1130** of Player C cannot be used by Player A to transport his army.

[0115] It should be recognized that the ghost controller **352** may apply different criteria for each player in determining whether a ghost is usable or the manner of using a ghost. For example, although the ghost **1130** of Player C may be usable by Player A when located in the proximity of a piece of land which is accessible by Player A, the same might not be true for Player B. Rather, a different set of rules may be applied to determine whether the ghost of Player C can be used by Player B.

[0116] Staying with this example, assume that the air force under the control of Player B can only attack the enemy within a limited range defined by the dotted circle **1150**. However, the ghost **1130** of Player C may be usable by player B to extend the attack range of Player B, when the ghost of Player C is located within the circle **1150**. For example, if the ghost **1130** of player C was located at Location Y within the circle **1150**, Player B may be permitted to land airplanes on the aircraft carrier of Player C, thus allowing the airplanes to refuel and extend the attack range of Player B to include the additional area covered by circle **1160**.

[0117] Another exemplary application of utilizing ghosts in a game **125** is provided with reference to FIG. **10**. As shown therein, a canvas **352** is provided for a detective game in which members of team **115** collaborate to solve mysteries or puzzles. Assume that the team **115** consists of Player C and Player D, and that only Player C is currently playing the game **125**.

[0118] When the avatar **1010** of Player C enters the room displayed by the canvas **352**, the canvas includes a door **1160** that has two keyholes **1030** and **1040**. The door **1060** can only be opened when the team **115** obtains two keys and inserts the keys into the two keyholes **1030** and **1040** on the door. Above the door **1060**, there is a vase **1050** located on a shelf. The vase **1050** contains the key needed to unlock one keyhole **1040**. A single user cannot reach the vase **1150** and get the key inside without the assistance of a teammate.

[0119] A ghost **1020** for Player D is inserted into the canvas **352** at a location adjacent to one of the keyholes **1030** indicating that Player D has unlocked the keyhole **1030**. The ghost controller **320** may configure the ghost **1020** for Player D to be usable in two different ways. First, the ghost **1020** of Player D can be utilized to boost the avatar **1010** of Player C to reach the vase **1050** and obtain the key hidden inside. After Player C obtains the key, the ghost controller **320** may also reconfigure to the ghost **1020** of Player D to be usable to assist Player C in opening the door **1060**.

[0120] As another example of using the ghost of teammate, consider a team that includes three users, each of which plays the game **125** in a different time period: past, present, future. That is, the three users are in the same place in the game, but in different times. The user in the future must reach the top of the mountain to accomplish a task. In order to reach the top of the mountain, the user from the past plants a tree and the user in the present waters that tree so that grows. As the user in the future plays the game, the user can see actions taken by his teammates (i.e., planting and watering the tree) which facilitate growth of the tree, and the user in the future can then climb the tree to reach the top of the mountain. In this manner, the future user interacts with the ghosts of teammates to accomplish the task.

[0121] In view of the above discussion, it should be recognized that the ghost controller **320** can be configured to handle anything related to providing a ghost in a particular game **125**. This may include determining how many ghosts are to be inserted, where the ghosts are to be inserted and how the inserted ghosts are to be configured. In certain games **125**, it may be preferable to only include certain types of ghosts. Likewise, in certain games **125**, it may be preferable to only include a single ghost for each player, while in other games **125** it may be preferred to include multiple ghosts for each player. Regardless of how ghosting features are implemented in a particular game **125**, the ghost controller **320** can be configured to handle all related operations.

[0122] FIG. **12** discloses a method **1200** for inserting a ghost feature into a canvas **352** for a game **125** that is being played by multiple users in accordance with an embodiment of the present principles. The method **1200** begins when a user loads the game **120** and engages in gaming activities during gameplay (block **1205**). As the user plays the game **125**, the user's client **110** records and stores data associated with the first user's gaming activities to the server (block **1210**).

[0123] The information may be utilized by the ghost controller **320** to determine the number of ghosts to be inserted into the canvas **352**, and the locations for inserting ghosts on the shared canvas **352** (block **1215**). For each ghost that is to be inserted into the canvas **352**, the ghost controller **320** may further determine how the ghost may be configured to be used by a particular player (block **1220**). As explained above, a ghost may be configured to be usable based on a variety of different criteria (e.g., based on the location of the ghost, based whether particular tasks or sub-tasks have been complete, etc.).

[0124] For every other player engaged in a gaming session (e.g., teammates or members of opposing teams **115**), the data stored on the server **120** is retrieved by a client **110** connected to the server **120** (block **1215**). The retrieved data is used to display the ghosts on the canvases **352** for the other players, and the ghosts are configured to be "usable" if applicable.

[0125] Although it is not necessary, the ghosting features described above may be advantageous in the case where members of team have to collaborate to accomplish a team task. As explained above, a "team task" generally refers a task that is subdivided into a plurality of subtasks that are completed by different members of a team **115**.

[0126] FIG. **13** is a diagram **1300** that illustrates how a game **125** may be configured to implement team tasks in accordance with an embodiment of the present principles. A team **115** is comprised of a plurality of team members X, Y through N.

[0127] User X receives or activates a new team task while playing a game **125**. As shown therein the team task is comprised of a plurality of subtasks (block **1310**). In this example, each subtask is assigned to a team member that is to complete the task. For example, subtask **1** is assigned to User X, subtask **2** is assigned to User Y, and it is shown that additional subtasks N can be assigned to additional members N of the team **115**. In certain embodiments, the sub-tasks may be executed in a particular order to complete the task.

[0128] After User X completes subtask **1** (block **1320**), a determination is made as to whether all of the subtasks have been completed (block **1370**). At this point, only one portion of the task (i.e., subtask **1**) has been completed. Therefore, a determination is made that there are still pending subtasks that have not been completed by the team **115** (block **1380**).

[0129] A notification may be sent to all or some of User X's teammates indicating the completion of the subtask (block **1385**). The notification may also indicate any remaining subtasks that are yet to be completed by the teammates. Notifications may be transmitted by the socializer **340** described above

[0130] When User Y subsequently plays the game **115**, User Y's client **110** retrieves data from the server **120** associated with the pending task (block **1330**). The retrieved data indicates that User X has completed subtask **1**, but the remaining subtasks are yet to be completed by User Y or the remaining team members N.

[0131] After User Y completes subtask **2** (block **1340**), a determination is made as to whether all of the subtasks have been completed (block **1370**). At this point, only two portions of the task (i.e., subtask **1** and subtask **2**) have been completed. Therefore, a determination is made that there are still pending subtasks that have not been completed by the team **115** (block **1380**). Once again, the socializer **340** or other component may transmit a notification to User Y's teammates to indicate that the subtask was completed and that pending subtasks still remain (block **1385**).

[0132] Thereafter, User N, which represents any remaining members of the team **115**, retrieves data associated with the task from the server **120** (block **1350**). The retrieved information indicates that subtasks **1** and **2** have been completed by Users X and Y, respectively, and that the remaining subtasks N has have not been completed.

[0133] User N completes subtask N (block **1360**). A determination is made as to whether any subtasks are pending (block **1370**). Since all subtasks have been complete, the task is determined to be completed (block **1390**). A notification is sent to all members of the team indicating that the task has successfully been completed (block **1395**).

[0134] FIG. **14** discloses a system **1400** for providing a joint inventory that is shared among a plurality of team members in accordance with an embodiment of the present principles. As shown therein, a virtual team **115** comprises a plurality of users that collaboratively play a game **125** together. As explained above, when a user plays the game **125**, the user may acquire virtual goods or objects that are incorporated into a joint or shared inventory **1420** for the virtual team **115**. The objects included in the joint inventory **1420** can be obtained individually by a single team member, or can be jointly acquired by collaboration between members of the team **115** (e.g., each player acquires a piece of an object and the object can be used when all subcomponents have been acquired).

[0135] An inventory manager **325** facilitates the joint ownership of objects in the joint inventory **1420**. The inventory manager **325** includes a variety of different components and associated functions for controlling objects stored in the joint inventory **1420**. Each of the components and functions associated with the inventory manager **325** may be utilized by any of the team members to manipulate objects in the joint inventory **1420**.

[0136] Specifically, the inventory manager **325** may include functionality permitting each of the users to add objects to the joint inventory **1420**. Once an object is added to the inventory **1420**, the object is commonly owned by all players on the team **115**, and each player can utilize the object as he or she pleases. The inventory manager **325** may also include functions for removing objects from the joint inventory **1420**. An object may be removed from the joint inventory **1420** in a variety of different situations, such as when the object has been used by a player or in response to explicit designation from a player to discard the object.

[0137] The inventory manager **325** may also include functions for permitting players or teammates to upgrade the status of an object in the joint inventory or to utilize objects in the joint inventory **1420**. For example, in certain games, a user can pay (e.g., using a virtual currency) to upgrade an object or an object may be upgraded based upon reaching new levels or accomplishing particular achievements.

[0138] Depending upon the particular object, the manner in which an object is utilized may differ. For example, a user can utilize a sword in the joint inventory **1420** by equipping the user's character with the sword and using the sword for battle, while a user can utilize a health elixir to restore the energy or life of the user's character.

[0139] In should be recognized that while inventory manager **325** generally permits players to manipulate the objects in the inventory without restriction, limitations may be placed on the users in certain embodiments. For example, in certain embodiments may prevent all players from deleting or removing a particular object stored in the joint inventory **1420**. In addition, while the inventory manager **325** depicted in FIG. **14** includes components for adding, deleting, upgrading and using objects in the shared inventory **1420**, the inventory manager **325** may be configured to handle any operation related to maintaining, managing or manipulating objects that are included in the joint inventory **1420**.

[0140] The figures in this disclosure are conceptual illustrations allowing for an explanation of the present invention. It should be understood that various aspects of the embodiments of the present invention could be implemented in hardware, firmware, software, or combinations thereof. In such embodiments, the various components and/or steps would be implemented in hardware, firmware, and/or software to perform the functions of the present invention. That is, the same piece of hardware, firmware, or module of software could perform one or more of the illustrated blocks (e.g., components or steps).

[0141] In software implementations, computer software (e.g., programs or other instructions) and/or data is stored on a machine readable medium as part of a computer program product, and is loaded into a computer system or other device or machine via a removable storage drive, hard drive, or communications interface. Computer programs (also called computer control logic or computer readable program code) are stored in a main and/or secondary memory, and executed by one or more processors (controllers, or the like) to cause

the one or more processors to perform the functions of the invention as described herein. In this document, the terms "machine readable medium," "computer program medium" and "computer usable medium" are used to generally refer to media such as a random access memory (RAM); a read only memory (ROM); a removable storage unit (e.g., a magnetic or optical disc, flash memory device, or the like); a hard disk; or the like.

[0142] Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including mobile telephones, PDA, pagers, hand-held devices, laptop computers, personal computers, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where local and remote computer systems, which are linked (either by hardwired links, wireless links, or by a combination of hardwired or wireless links) through a communication network, both perform tasks. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0143] Notably, the figures and examples above are not meant to limit the scope of the present invention to a single embodiment, as other embodiments are possible by way of interchange of some or all of the described or illustrated elements. Moreover, where certain elements of the present invention can be partially or fully implemented using known components, only those portions of such known components that are necessary for an understanding of the present invention are described, and detailed descriptions of other portions of such known components are omitted so as not to obscure the invention. In the present specification, an embodiment showing a singular component should not necessarily be limited to other embodiments including a plurality of the same component, and vice-versa, unless explicitly stated otherwise herein. Moreover, applicants do not intend for any term in the specification or claims to be ascribed an uncommon or special meaning unless explicitly set forth as such. Further, the present invention encompasses present and future known equivalents to the known components referred to herein by way of illustration.

[0144] The foregoing description of the specific embodiments so fully reveals the general nature of the invention that others can, by applying knowledge within the skill of the relevant art(s) (including the contents of the documents cited and incorporated by reference herein), readily modify and/or adapt for various applications such specific embodiments, without undue experimentation, without departing from the general concept of the present invention. Such adaptations and modifications are therefore intended to be within the meaning and range of equivalents of the disclosed embodiments, based on the teaching and guidance presented herein.

[0145] While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example, and not limitation. It would be apparent to one skilled in the relevant art(s) that various changes in form and detail could be made therein without departing from the spirit and scope of the invention. Thus, the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What is claimed is:

1. A method for asynchronously providing a game for play by multiple players as a team over a network, the method comprising:

storing shared elements of an asynchronous game on a computer readable storage medium that is accessible to the players of the team over the network;

receiving data from a first player of the team indicative of gaming activities modifying the shared elements of the game;

updating, with a processor, the shared elements with the data received from the first player to generate an updated game state; and

transmitting update data representing the updated game state to at least one other player on the team.

2. The method as recited in claim **1**, further comprising storing a team task which includes a plurality of subtasks that are performed by at least two separate members of the team in order to complete the team task.

3. The method as recited in claim **2**, further comprising assigning at least one of the subtasks to a player on the team based on a unique role that is assigned to the player, said role being the only role that can accomplish the subtask.

4. The method as recited in claim **1**, further comprising merging the update data into a single presentation for display to the at least one other player during gameplay.

5. The method as recited in claim **1**, further comprising utilizing a jointly owned object in the shared elements that was previously acquired by at least one other player of the team.

6. The method as recited in claim **1**, wherein the shared elements include a shared canvas and the method further comprises modifying the shared canvas in accordance with gaming activities of the first user and including data indicative of the first user's gaming activities in the update data.

7. The method as recited in claim **1**, further comprising determining ghosting information to be transmitted with the update data to the at least one other player to indicate gaming activities engaged in by the first player.

8. The method as recited in claim **7**, further comprising utilizing the ghosting information to display a ghost associated with the first player to the at least one other player and configuring the ghost to be usable by the at least one other player.

9. The method as recited in claim **1**, wherein the shared inventory includes at least one jointly acquired object that is obtained by a collaborative effort of at least two players on the team.

10. The method as recited in claim **1**, wherein the team represents a group of players that play together in a plurality of different games.

11. A system for providing an asynchronous game to multiple players on a team over a network, comprising:

a database for storing a shared elements of an asynchronous game that is accessible to the players of the team over the network; and

a server configured to:

receive data from a first player of the team indicative of gaming activities modifying the shared elements;

update the shared elements with the data received from the first player to generate an updated game state; and

transmit update data representing the updated game state to at least one other player on the team.

12. The system as recited in claim **11**, further comprising a team controller, being stored on the server, which is configured to store information relating to a team task on the computer readable storage medium, the team task including a plurality of subtasks that are performed by at least two separate members of the team in order to complete the team task.

13. The system as recited in claim **12**, wherein the team controller is further configured to assign at least one of the subtasks to a player on the team based on a unique role that is assigned to the player, said role being the only role that can accomplish the subtask.

14. The system as recited in claim **11**, further comprising a merger for merging the update data into a single presentation for display to each user during gameplay.

15. The system as recited in claim **11**, wherein each player on the team can utilize the at least one jointly owned object in the shared elements that was previously acquired by at least one other player.

16. The system as recited in claim **11**, further comprising a ghost controller configured to determine ghosting information for the at least one other player to indicate gaming activities engaged in by the first player.

17. The system as recited in claim **16**, wherein the ghosting information is utilized to display a ghost associated with the first player to the at least one other player, and to configure the ghost for use by the at least one other player.

18. The system as recited in claim **11**, wherein the shared elements include an inventory having at least one jointly acquired object that is obtained by a collaborative effort of at least two players on the team.

19. The system as recited in claim **11**, wherein the team represents a group of players that play together in a plurality of different games.

20. A non-transitory computer storage medium comprising a computer readable program for providing an asynchronous game to multiple players on a team over a network, wherein the computer readable program when executed on a computer causes the computer to:

provide a shared elements to the players of the team over the network;

receive data from a first player of the team indicative of gaming activities modifying the shared elements;

update the shared elements with the data received from the first player to generate an updated game state; and

transmit update data representing the updated game state to at least one other player on the team.

* * * * *