GAME USER APPARATUS

Inventor: Uwe PROCHNOW, Essen (DE)

Correspondence Address:
MCDERMOTT WILL & EMERY LLP
18191 VON KARMAN AVE., SUITE 500
IRVINE, CA 92612-7108 (US)

Assignee: GDI Game Domain International PLC, Southhall (GB)

Appl. No.: 12/407,753
Filed: Mar. 19, 2009

Foreign Application Priority Data
Mar. 20, 2008 (GB) 0805196.3
Dec. 19, 2008 (GB) 0823231.6

ABSTRACT
A user game apparatus configured to facilitate game playing according to which first and second separate supplies of game data are received from a remote game distribution server and combined during game play responsive to game playing events, the apparatus comprising: a request module arranged to receive a first supply of data from a first data path and a second supply of data from a second data path and operable to combine said first and second data to produce a natural game play experience; and a synchroniser configured to control timing of supply of the first supply of data and the second supply of data from respective data buffers.
Select game

Receive game engine (if required)

Receive a first list of required essential minimum data

Compare the first list with the passive data stored in the passive data storage device

Determine what required essential minimum passive data is not stored in the passive data storage device

Transfer a second list of the required essential minimum passive data to the game server

Receive the second list of essential minimum passive data and associated references

Store the received essential minimum passive data in the passive data storage device

FIGURE 3
GAME USER APPARATUS

FIELD

[0001] The invention relates to the field of gaming. More specifically, an embodiment of this invention relates to a game user apparatus for enabling a user to play games, methods and software for deploying games.

BACKGROUND

[0002] Digital distribution is the principle of providing digital content over the Internet, and other networks, in the form of products or services. With broadband internet becoming evermore ubiquitous, the digital distribution of all types of media content is becoming increasingly desirable. Content delivery networks consisting of systems of servers and desktop computers have been deployed to deliver digital content to end users operating for example desktop PCs or consoles across the Internet.

[0003] One existing platform of this type for example is Valve Corporation’s “Steam” product, which is a digital distribution, multiplayer and communications suite. It is primarily used to digitally distribute and manage various kinds of computer games, ranging from first-person shooters and RPGs (role playing games) to racing games.

[0004] Systems such as this allow users to purchase and acquire games access through a digital distribution network. In this regard, instead of receiving a box with a CD/DVD, purchased software is distributed by a server to the user, for example, via a pre-registered user account through which the software can be accessed and downloaded to a local client running on the user’s computer.

[0005] Known digital distribution platforms have operated with a distributed file system. The distributed file system allows a game to launch before it has been completely downloaded locally on to the player’s computer. Most typically, this has been done by creating lists of essential game files and having a pre-loader request them from the distribution server only when they are needed or are about to be needed. According to these systems, it has been possible for a linear game to begin playing with only the executable code and a buffer of the first few areas downloaded. Such systems have been problematic up to now for various reasons. One major problem is that the game play will generally hang whilst data downloads in the background because modern games are many hundreds or even thousands of megabytes in size. Another problem is that downloads are vastly slowed down on slower connections or when the service providers get busy and are less able to cope with providing high bandwidth downloads. This problem is increased further for non-linear games, where the possibilities for gameplay are vast and the data required is always changing. For these reasons, this kind of content delivery has not been used extensively to date, since it has been impractical and frustrating for the end user.

[0006] The present invention aims to provide an improved content delivery system for computer games. More specifically, the present invention provides an application which allows a user to receive game data at their computer quickly and securely over a network such as the internet.

SUMMARY

[0007] According to one embodiment of the invention, there is provided a user game apparatus configured to facilitate game playing according to which first and second separate supplies of game data are received from a remote game distribution server and combined during game play responsive to game playing events. The user game apparatus comprising: a request module arranged to receive a first supply of data from a first data path and a second supply of data from a second data path and operable to combine said first and second data to produce a natural game play experience; and a game synchroniser configured to control timing of supply of the first supply of data and the second supply of data from respective data buffers.

[0008] According to another embodiment of the invention, the first data has an average file size smaller than an average file size of the second data.

[0009] According to one embodiment of the invention, there is provided a user game apparatus configured to facilitate game playing according to which active game data and passive game data are received from a remote game distribution server and combined during game play responsive to game playing events. The user game apparatus comprising: a first module arranged to receive a supply of active data from an active data path and a supply of passive data from a passive data path, said first module being operable to combine said active and passive data to produce a natural game play experience; and a game synchroniser configured to control timing of supply of the active data from the active data path to the first module and the supply of the passive data from the passive data path to the first module.

[0010] According to another embodiment of the invention, the user game apparatus further comprises: a real-time delivery module supplying active data from an active data buffer responsive to the control of the game synchroniser.

[0011] According to another embodiment of the invention, loading of the active data buffer is controlled by an active data manager communicating with the remote game distribution server.

[0012] According to another embodiment of the invention, the first module is configured to supply game play event indicators along the active data path.

[0013] According to another embodiment of the invention, game play event indicators are supplied via the active data path to a remote game distribution server, wherein the active data manager receives from the remote game distribution server active data required for future game play.

[0014] According to another embodiment of the invention, the real-time delivery module is configured to process the game play event indicators to determine what active data is required for future game play, and the active data required for future game play is requested from the remote game distribution server.

[0015] According to another embodiment of the invention, the active data required for future game play is staged into the active data buffer ahead of being required by the first module.

[0016] According to another embodiment of the invention, the active data required in future game play is staged into the active data buffer in dependence upon the game play event indicators dictated by the game code.

[0017] According to another embodiment of the invention, the active data stored in the active data buffer is deleted once used.

[0018] According to another embodiment of the invention, the active data stored in the active data buffer is deleted after a predetermined period of time; is deleted after the active data buffer capacity is exceeded; and/or is deleted after implementation in game play.
According to another embodiment of the invention, the user game apparatus further comprises: a game constructor module operable to construct and run a clone version of the game that is substantially complete save for the absence of active data.

According to another embodiment of the invention, the first module is configured to supply game play event indicators along the passive data path.

According to another embodiment of the invention, the game play event indicators are supplied to the remote game distribution server, and wherein a passive data manager receives from the remote game distribution server an indication of what passive data is required for future game play.

According to another embodiment of the invention, a recall manager is configured to process the game play event indicators to determine what passive data is required for future game play, and to transfer to a passive data manager an indication of what passive data is required for future game play.

According to another embodiment of the invention, the passive data manager is configured to determine if the passive data required for future game play is available in a passive data store.

According to another embodiment of the invention, the passive data is operable to request the passive data required for future game play from the remote game distribution server and to store the requested passive data in the passive data store in the event it is not already available in the passive data store.

According to another embodiment of the invention, each passive data object associated with a unique reference.

According to another embodiment of the invention, the passive data comprises a plurality of passive data objects, each passive data object associated with a unique reference, and wherein the indication of what passive data is required for future game play comprises a plurality of unique references.

According to another embodiment of the invention, the passive data comprises a plurality of passive data objects, each passive data object associated with a unique reference, and wherein the indication of what passive data is required for future game play comprises a plurality of unique references and the request for passive data from the remote game distribution server comprises a plurality of unique references.

According to another embodiment of the invention, the passive data path further comprises a passive data buffer from which passive data may be transferred to the game constructor module.

According to another embodiment of the invention, the passive data is supplied from the passive data store to said game constructor module via the passive data buffer.

According to another embodiment of the invention, the passive data required in future game play is staged into the passive data buffer in dependence upon game play events as dictated by the game code.

According to another embodiment of the invention, an amount of active data stored in the active data buffer is less than an amount of passive data stored in the passive data buffer.

According to another embodiment of the invention, the active data comprises a plurality of active data objects with file sizes that are on average smaller than the average file size of passive data objects.

According to one embodiment of the invention, there is provided a method of operating user game apparatus provided at a user terminal and configured to facilitate game playing. The method comprising: receiving a supply of active data from a remote game distribution server and a supply of passive data from a remote game distribution server; controlling timing of the supply of the active data and the supply of the passive data; and combining said active and passive data during game play responsive to game playing events to produce a natural game play experience.

According to another embodiment of the invention, the method further comprises: supplying game play event indicators to the remote game distribution server; and receiving from the remote game distribution server active data required for future game play.

According to another embodiment of the invention, the method further comprises: supplying game play event indicators to the real-time delivery module; processing the game play event indicators to determine what active data is required for future game play; and requesting the active data required for future game play from the remote game distribution server.

According to another embodiment of the invention, the method further comprises: supplying the active data required for future game play into an active data buffer.

According to another embodiment of the invention, the method further comprises: deleting the active data from the active data buffer once used.

According to another embodiment of the invention, the method further comprises: deleting the active data from the active data buffer after a predetermined period of time; after the active data buffer capacity is exceeded; and/or after implementation in game play.

According to another embodiment of the invention, the method further comprises: constructing and running a clone version of the game that is substantially complete save for the absence of active data.

According to another embodiment of the invention, the method further comprises: supplying game play event indicators to the remote game distribution server; and receiving from the remote game distribution server an indication of what passive data is required for future game play.

According to another embodiment of the invention, the method further comprises: supplying game play event indicators along the passive data path to a recall manager; and processing the game play event indicators to determine what passive data is required for future game play.

According to another embodiment of the invention, the method further comprises: determining if the passive data required for future game play is available in a passive data store.

According to another embodiment of the invention, the method further comprises: requesting passive data from the remote game distribution server in the event it is not already available in the passive data store; and storing the requested passive data in the passive data store.

According to one embodiment of the invention, there is provided a computer program product comprising programme code means for performing the method described above.

According to one embodiment of the invention, there is provided a computer readable medium recorded with computer readable code arranged to cause a computer to perform the method described above.

According to one embodiment of the invention, there is provided a computer programme code means for performing the method described above.
DESCRIPTION OF THE DRAWINGS

[0047] For a better understanding of the invention and to show how the same may be carried into effect, reference will now be made by way of example to the accompanying drawings:

[0048] FIG. 1 illustrates schematically a gaming system;

[0049] FIG. 2 illustrates schematically game user apparatus;

[0050] FIG. 3 illustrates a flow diagram of a process for downloading data from a game server to a user’s device;

[0051] FIG. 4 illustrates schematically a buffer zone; and

[0052] FIG. 5 illustrates schematically several buffer zones.

DETAILLED DESCRIPTION

[0053] Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings.

[0054] FIG. 1 illustrates the main components of a gaming system 1. The gaming system 1 comprises game conversion module 10, game server 20 and user devices 30. The user devices 30 are connected to the game server 20 over a network such as the internet 40.

[0055] The game conversion module 10 converts a game of known format into a format suitable for use in the gaming system 1. The game conversion module 10 then transfers the converted game (in one embodiment a copy of the converted game) to the game server 20.

[0056] The game conversion module 10 may not have a permanent connection to the game server 20. Therefore, following conversion of a game, the conversion module 10 establishes a connection with the game server 20, transfers the converted game to the game server 20 and then terminates the connection. This is illustrated by the dotted line in FIG. 1. Alternatively, the game conversion module 10 may have a permanent connection to the game server 20. In either example the connection may be a wired or wireless connection. In addition, the game conversion module 10 may not have a connection to the game server 20. In this instance, following conversion of a game, the game is provided to the game server 20 via a storage medium such as a CD, DVD or memory stick etc.

[0057] Further details of the conversion module 10 can be found in co-pending patent application entitled: “Apparatus and Methods for Game Conversion” filed by the same applicant as this application, although this is not required to understand and implement the present invention.

[0058] In brief, in order to convert a game of known format into a format suitable for use with the gaming system 1, the game conversion module 10 converts the game into active data elements and passive data elements, such that the whole game can be transferred to the game server 20 as either active data or passive data.

[0059] Typical examples of passive data are the elements of textures, mp3 files, wave files, text files and such like. However, broadly speaking, passive data is any data which can be downloaded to and stored locally on a user’s device 30, and can potentially be replaced by a reference identifier. The game engine (for example: Unreal engine, Torque engine, Quake engine, Half Life engine) can also be classed as passive data because it is possible to use a version of the game engine stored locally on a user’s device 30.

[0060] Active data is supplied to the user’s device 30 running the game in substantially real time or close to it. Active data is usually data which is required in order to give the passive data behaviour during game play. Potentially, any type of data can be classified as active data. However, according to one example, it is the game object data (for example parameter data) which is classified as active data. To this effect, in order for there to be a composition of all the basic data components in the game to create the sophisticated, interactive virtual object (or game token) as seen by the player during game play, the passive data (e.g., the textures, the game engine, etc.) is combined with the active data (parameters describing its behaviour). Without the active data, therefore, the game cannot be played because the passive data will not compile/ render/ behave properly on the user’s device 30 due to the fact that there is no data describing how the game elements are combined and how they interact with each other in the game environment. The active data is downloaded separately from the passive data to the user’s device 30, and combined with locally stored passive data in order to create the playable game. According to one example, the active data and passive data may be combined on-the-fly upon separate arrival at the user’s device 30.

[0061] Other attributes of active data include: active data is deleted from the user’s device 30 once it has been used in game play or after a predetermined period of time; active data tends to have a small file size relative to the passive data; and active data is buffered with a relatively short lead time in comparison to passive data.

[0062] Certain types of game object data are relatively small when compared with other data elements such as textures, audio files and the game engine, which can all have very large file sizes. It is therefore advantageous to have the object data as active data for two reasons. Firstly, having the object data as active data operates as a form of digital rights management in that the game cannot be played by a user without the necessary active data to compose the game play. Therefore active data is only stored momentarily on a user’s device 30 and must be downloaded from the server 20 separate from the passive data. Secondly, the object data (generally active data) being so small means that games can be distributed using only a limited amount of network bandwidth, with all (or at least a large majority) of the large data (generally passive data) files being stored locally on the user’s device 30.

[0063] According to one example, a site threshold is used to define data suitable for categorisation as active data. For example, any data file smaller than 256 kilo is selected as potentially active data. Then a predetermined percentage of that selection, e.g. 40%, is classified as active data (the other 60% being classified as passive data).

[0064] As stated above, the passive data can be replaced by a reference. In conventional games an element of data, for example the texture grass may appear numerous times throughout the game. In order to convert a conventional game into a format of the gaming system 1, the game conversion module 10, identifies all the instances where an element of passive data appears more than once (either identical or similar), by checking the data content of each file, associates an unique reference with one of the element of data files and replaces all further instances of the element of data file with the unique associated reference (deleting all the further instances of the element of data file). For example the determination may be made pixel by pixel in the case of a texture, however, any suitable technique well known in the art may be
used to assess similarity of the file contents as required. This greatly reduces the size of the game (for example a game of known format may be 1.2 GB but following conversion is reduced to 190 MB).

[0065] For example, the texture grass may appear 100 times in a game, in numerous levels, each texture grass file looking the same (identical or similar) in a game but having different file names. The game conversion module 10 identifies each instance when the texture grass appears, associates an unique reference with one instance of the texture grass, and replaces the remaining instances with the associated unique reference. Therefore, the game conversion module 10 reduces the size of the game by deleting repeated (and thus redundant) data. This results in a converted game which is of the same quality as a non-converted game, since all necessary source code of the game is not changed, but is of a reduced size. Consequently, the look, feel and game play is not altered by the game conversion module 10.

[0066] The game conversion module 10 also determines a game essential minimum. The essential minimum identifies the minimum amount of data required in order for a user to begin playing the game, for example, the minimum amount of data to enable the game to start up, or to start up and load the first level, or a portion of the first level or an otherwise fully playable portion of the game. The essential minimum is comprised entirely of passive data. Therefore, any data which may have been considered active data, due to its content or size is considered passive data if it is part of the essential minimum. Following download of the essential minimum to a user's device 30, the user can begin playing the game (for a limited period of time). In another embodiment, the user may need a supply of active data in real time or substantially real time in order to play the game at all. In either case, in order to continue playing the game, further active and passive data will be required and, accordingly, the user's device 30 requires a connection to the game server 20.

[0067] Each converted game which is output from the game conversion module 10 may be identifiable by a ContainerID. The ContainerID comprises a GameID with an additional suffix describing game packets. The GameID is a unique code to identify the game throughout the entire gaming system 1. According to one embodiment, the unique code is an alphanumeric code, e.g. two letters and five digits in length, optionally with a region suffix (e.g. “DE” for Germany, “UK” for United Kingdom etc.) indicating the language version of the game. Note that each language version of the game will generally undergo a separate conversion process, although in some circumstances, this may not be necessary. The additional suffix describing game packets may be a four digit code. For instance, the GameID “TR000001DE” plus the ContainerID suffix “0109” may represent the first packet (01) of the German language version of Tomb Raider 1, which has a total of 9 packets (09). In this context, the first packet “01” is the essential minimum as described above.

[0068] These Containers may contain both active and passive data for a game. Alternatively, there may be corresponding containers and packets for each of the active and passive data. In any case, the passive data and associated references of the converted game and the active data of the converted game are stored or labelled separately so they can be deployed to the client separately.

[0069] Following conversion of a game, the game conversion module 10 stores all the passive data and associated references of the converted game in a passive data storage device (not illustrated) and stores all the converted active data of the converted game in an active data storage device (not illustrated).

[0070] In one embodiment the passive data is divided into categories for storage. For example, all textures are stored in one table, all mp3 files are stored in another table, all wave files are stored in another table etc. Therefore, the passive data is stored in one storage device but the storage device is divided into separate tables based on the category of passive data.

[0071] The passive data storage device and the active data storage device of the game conversion module 10 store all the converted games which have been converted by the game conversion module 10.

[0072] The same element of passive data may appear in numerous games. For example, the texture grass may appear in five different games. Therefore, in order to avoid the duplication of passive data within the game conversion passive data storage device, the game conversion module 10 also identifies all the instances where an element of passive data within the game being converted appears (either identical or similar) in the passive data storage device. Therefore, if an element of passive data which appears in a game being converted is already in the passive data storage device, then the unique reference associated with the element of passive data in the passive data storage device is used to replace all instances of that element of passive data within the game.

[0073] Taking the example above of the texture grass appearing in numerous different games, when a new game is to be converted, the game conversion module 10 identifies all instance of the texture grass within the new game, compares the texture grass with the data stored in the game conversion passive data storage device, determines that the texture grass is already stored in the game conversion passive data storage device, identifies the unique reference which is associated with the texture grass in the game conversion passive data storage device, and replaces all instances of the texture grass within the new game with the unique reference.

[0074] Each passive and active data element may comprise metadata provided by the game conversion module 10. The metadata may comprise:

[0075] FileID—a unique code to identify the file throughout the entire gaming system 1. According to one embodiment, the unique code is an alpha-numeric code, e.g. two letters and ten digits in length.

[0076] FileExtension—a record containing the given name of the file including its extension e.g. “.tcl”, “.png” “.jpg” “.wav” etc.

[0077] FileTypeID—a record indicating the file extension, e.g. “.bmp”, “.jpg”, “.mp3”, “.wav” and such like. According to one embodiment, proprietary file extensions are standardised during the conversion process. For instance, a standard “.jpg” file may be renamed as “.abc” by a given game developer for a number of reasons. This process may be reversed so that all standard files have the correct, accepted file extension format. According to one embodiment, file extensions are standardised according to a labelling system representing individual file types. For instance, the following numbering system may be employed: 00100 representing the general category of “graphics files” with 00101−jpeg files, 00102−bmp files, 00103−png files etc; and 00400 representing “sound files” with 00401−wav files, 00402−ogg files, 00403−mp3 files etc.
FileLink—is a marker indicating the position of the file within the file structure of the installed game. In other words, it is a route indicator.

GameID—a unique code to identify the game throughout the entire gaming system 1. According to one embodiment, the unique code is an alpha-numeric code, e.g., two letters and five digits in length, optionally with a regional suffix (e.g., "DE" for Germany, "UK" for United Kingdom etc.) indicating the language version of the game. Note that each language version of the game will generally undergo a separate conversion process, although in some circumstances, this may not be necessary.

FileSize—the size of the file, for example in kilobytes

ReferenceStatusID—made up of three sub-meta-data flags: “No” (0), “Yes” (1) and “Not Tested Yet” (2), each indicating whether or not the file is replaceable by a file already present in the game conversion database.

ReplaceFileID—if the ReferenceStatusID=1, then the ReplaceFileIDs comprises a list of all other IDs which the file is a reference file to. In other words, it comprises a list of all equivalent files that can be replaced by the present file.

ClassID—is made up of two sub-meta-data flags: an indicator of whether the file is classified as active (0) or passive (1).

SubClassID—is a locking flag which prevents an active file being replaced by another file. Accordingly, a file may be marked “locked” (0) or “replaceable” (1). This feature enables the prevention, for example, of a certain file being replaced based on performance considerations, and prevents collision between ClassIDs.

The game conversion passive data storage device at the game conversion module 10 comprise all the passive data required for all the games which have been converted by the game conversion module 10 and the game conversion active data storage device at the game conversion module 10 comprise all the active data required for all the games which have been converted by the game conversion module 10. The passive data storage device is essentially a shared library of passive data.

Further details of the game server 20 can be found in co-pending patent application entitled: “Game Server” filed by the same applicant as this application, although this is not required to understand and implement the present invention.

In brief, the game server 20 comprises converted game data converted by the game conversion module 10. The game server 20 transfers the converted game data to the user’s device 30 as required.

In order for the user to play a converted game of the gaming system 1 a game user apparatus (game processing application) may be provided on the user’s device 30.

An user can access the game server 20 over the internet from an user’s device 30, in order to play games, and if required set up a gaming account with the gaming server 20. The user accesses the server 20 via a web interface. Upon first use the user may be required to set-up an user account with the gaming server 20, for example, the user may be required to enter data such as, their name, address, date of birth, financial account details etc. A user account is then created for each user by the gaming server 20, such that the user can access the game server 20 and play any of the converted games which are stored at the gaming server 20.

Upon first use of the gaming server 20, the gaming server 20 determines whether the user’s device 30 comprises the game user apparatus required in order to play games converted by the game conversion module 10. If it is determined that the user’s device 30 does not comprise the game user apparatus, then the gaming server 20 downloads the game user apparatus to the user device 30.

In addition upon first use, the gaming server 20 identifies the hardware environment of the user’s device 30, such as the graphics card the user’s device 30 comprises and the processing speed of the CPU (central processing unit) of the user’s device 30.

The game server 20 requires a two-way interactive communication with the user’s device 30 during game play. This is because during game play, not only is it necessary for the game server 20 to download data (active and passive) to the user device 30, but it is also necessary for the user’s device 30 to send data to the game server 20, such as events occurring during game play and information about decisions made by the game player (user) during game play.

Therefore, the gaming server 20 also determines the ping time and bandwidth of the user’s connection upon first (and subsequent) use of the gaming server 20. The ping time and bandwidth of the user’s connection affects the speed with which the user’s device 30 gets a response from the game server 20 when the user’s device 30 sends a request for data to the game server 20.

In one embodiment, the hardware environment of the user’s device 30 and/or the ping time and bandwidth of the user’s device 30 is checked at predetermined intervals, for example each time the user connects to the game server 20, or each time the user selects a new game for playing. In addition, the hardware environment of the user’s device 30 and/or the ping time and bandwidth of the user’s device 30 may be checked throughout game play.

In order to avoid pauses during game play whilst data is being downloaded from the game server 20, the server 20 may alter the buffer zone (explained in further detail below) for each user as a result of their hardware environment, ping time and bandwidth. Users with poor ping time and bandwidth are assigned a larger buffer zone than users who have good ping time and bandwidth.

Following set-up of an user account, the user becomes an authorised user. In one embodiment, an authorised user may be required to enter an user name and password in order to gain access to the game server 20. The gaming server 20 then compares the entered user name and password with authorised user data held in an user database (not illustrated) in order to verify if the user is an authorised user. Once it is confirmed that the user is an authorised user, the user is allowed to access the game server 20 and play games.

In one embodiment, the gaming server 20 may store user account data. For example, a user may be required to pay in order to play a game at the game server 20. Therefore, the user account data may be data regarding whether the user has sufficient funds in order to play a game at the game server 20. If it is determined that the user does not have sufficient funds in their account, then access to game play via the game server 20 may be denied until sufficient funds have been added to the user’s account.

Following confirmation that the user has sufficient funds, the user can select any one of the converted games stored in the game server 20 to play. The gaming system 1 of
the invention enables a user to play a game over the internet, with very short download times.

In one embodiment, the gaming server 20 may require the user to be an authorised user and have sufficient funds in their account before enabling a user to play a game. In another embodiment, the user may not be required to be an authorised user or to have sufficient funds in their account before commencing game play.

A detailed description of the components of the game user apparatus 300 will now be provided with reference to FIG. 2.

The game user apparatus 300 comprises a passive data storage device 302. In one embodiment, the passive data storage device 302 is initially empty. In another embodiment, initial passive data may be transferred with the game user apparatus 300 from the game server 20 to the user device 30 in order to populate the passive data storage device 302. In another or additional embodiment, passive data may be provided on a CD (compact disc)/DVD/memory stick etc. which the user obtains in order to populate the passive data storage device 302.

In order to play the selected game the essential minimum data (discussed above) is downloaded from a passive data storage device at the server 20 to the passive data storage device 302. In addition the connection between the server 20 and the user’s device 30 must be active, so that active data and further passive data can be downloaded to the user’s device 30 as required during game play.

As stated above the game engine is considered a special type of passive data. Upon user selection of a game for playing, the server 20 determines whether the user’s device 30 has the required game engine, and if the user’s device 30 does not have the required game engine, then it is downloaded to the passive data storage device 302. The game is then run on this locally stored game engine which combines the necessary active data received from the game server 20 and the necessary passive data downloaded to the passive data storage device 302 from the game server 20 and retrieved from the passive data storage device 302 when required. If the user’s device 30 does have the required game engine, then data setting the parameters of the game engine, to configure the game engine, is downloaded to the passive data storage device 302.

FIG. 3 illustrates the process of the download of passive data from the game server 20 to the passive data storage device 302 at the user’s device 30.

When an user selects a game for playing (at step S200), the users selection is transferred via a communication interface 32 of the user’s device 30 to the game server 20. The game server 20 determines what game engine is required to play the selected game and checks whether the user’s device 30 has this game engine. As stated above, the game engine is considered to be passive data. Therefore, in order to determine if the game user apparatus 300 comprises the required game engine, the game server 20 informs the passive data manager 304 what game engine is required. The passive data manager 304 transfers this request to the passive data storage device manager 306, which checks whether the required game engine is stored in the passive data storage device 302.

If the user’s device 30 does not have the required game engine, then the game engine is transferred to the user’s device 30 (step S202) for storage in the passive data storage device 302. The passive (and active) data is transferred to the user’s device 30 via the communication interface 32.

In addition to determine what game engine is required by the game, the game server 20 also determines what administrative features, such as physics engine, control application etc. are required by the game and transfers those which are required to the user’s device 30. In one embodiment, all the administrative features of the game are downloaded to the user’s device 30 at the beginning of the game download.

The game server 20 also determines the essential minimum data required in order to play the selected game. The user device 30 then receives a first list of unique references indicating the essential minimum data which is required for the selected game from the game server 20 (step S204).

As stated above, the user device 30 comprises a passive data manager 304. The passive data manager 304, in combination with the passive data storage device manager 306, compares the first list of unique references with the unique references associated with passive data already stored in the passive data storage device 302 (step S206) and determines which of the first listed passive data is not already stored in the passive data storage device 302 (step S208). The passive data manager 304 is capable of two way communication (request and response communication) with the passive data storage device manager 306.

As a result of the references, a quick comparison can be performed between the first list of unique references and the unique references associated with the passive data held in the passive data storage device 302. Consequently, only the unique references associated with the passive data (in the form of a first list) are received from the game server 20. Therefore, the amount of data which is transferred from the game server 20 to the user device 30 is very small and thus enable quick processing of the data.

The passive data manager 304 then returns a second list of unique references to the game server 20 (step S210) via the communication interface 32. The second list indicates which of the first listed passive data is not stored in the passive data storage device 302 and thus is required by the passive data storage device 302, so that the user can begin playing the selected game.

In one embodiment, if the user device 30 does not have any data stored in the passive data storage device 302, then the second list of references is identical to the first list of references. However, if some of the required passive data is already stored in the passive data storage device 302, then this passive data does not need to be transferred from the game server 20 to the passive data storage device 302 of the user’s device 30. Consequently, the second list of references is likely to be smaller than the first list of references.

The game server 20 uses the second list of references to determine what passive data and its associated reference is required to be transferred to the user’s device 30. The user device 30 then receives a copy of the required passive data and its associated reference from the game server 20 (step S212). Therefore, only new passive data (passive data not already stored in the user passive data storage device) is transferred to the user device 30. This process reduces that amount (size) of data which needs to be transferred from the game server 20 to the user’s device 30 in comparison to conventional systems where the entire new game data is transferred.

The passive data storage device manager 306 then stores the received passive data and associated reference in
the passive data storage device 302 (step S214), by updating
the passive data storage device 302 with the downloaded
passive data. Management of where the passive data is to be
stored in the passive data storage device 302 is controlled by
the passive data storage device manager 306.

[0116] In one embodiment, the passive data storage device
302 stores the passive data based on categories, such that all
textures are stored in one table, all mp3 files are stored in
another table, all wave files are stored in another table etc.
Therefore, the passive data storage device 302 is one storage
device storing all the passive data but is divided into separate
tables based on the category of passive data. In this embodi-
ment, the passive data storage device manager 306 deter-
nines the table in which each piece of passive data should be
stored.

[0117] In another embodiment, the passive data may also
be associated with a reference to the game the passive data is
required for, and/or the level of the game the passive data is
required for etc, and/or the genre of game the passive data is
required for etc. In one embodiment, the association of such
further information with the passive data can be used to
decrease the download time from the game server 20. For
example, whenever data is to be downloaded, a comparison is
first undertaken between the references of the passive data to
be downloaded and the references of the passive data already
stored in the passive data storage device 302 (step S206). If
the genre of the game is known, then the comparison may first
be carried out on the references of passive data already stored
in the passive data storage device 302 which is in the same
genre. For example, the texture “tarmac” for a racing track is
likely to be used in most racing games, therefore, if the
passive data manager 304 knows that the new game is a racing
game, then the comparison can first be carried out on the
references of passive data already stored in the passive data
storage device 302 which are used in racing games.

[0118] Since only new passive data (passive data not
already stored in the passive data storage device 302) is
transferred to the user device 30, the more games that the user
has selected for playing, the more passive data will be stored
in the passive data storage device 302 at the user’s device 30.
Thus, the amount of passive data required to be transferred
from the game server 20 to the user device 30 should be
reduced with each game which is selected and downloaded,
such that the time required to download the passive data to
the user’s device 30 decreases with each new game. For example,
if the user has previously played (and thus downloaded) the
game Tomb Raider™ and then selects the game Tomb Raider II™,
since it is highly likely that Tomb Raider II™ uses a lot of
the same passive data as Tomb Raider™, the amount of
passive data which needs to be downloaded for Tomb Raider
II™ is reduced. However, this may not be the case, for
example, if the user has previously only played sports games
and then decides to select a role playing game, which is
unlikely to use the same passive data.

[0119] In addition the download time is affected by the ping
time, bandwidth and hardware of the user’s device 30. Con-
sequently, the essential minimum data required for initial
download may be supplemented with further initial download
data (if required) depending on the determined ping time and
bandwidth of the connection to the user’s device 30 (as dis-
cussed above). For example, if the users connection is slow,
the game content processing module 120 may determine that
the user’s device 30 requires the essential minimum data and
further additional active and passive data to be downloaded to
the user’s device 30 before game play can commence. In this
instance the extra passive data is transferred together with or
separately from the essential minimum data using a similar
process to that described in FIG. 3. The extra active data may
also be transferred together with or separately from the
essential minimum data. However, since the active data cannot be
replaced by a reference and is not permanently stored at the
user’s device 30, no comparison (such as steps S210 to S218)
is required.

[0120] The entire game is not required to be downloaded
before game play begins, only the essential minimum is
required. The game data (active and passive data) is then
downloaded in the background during game play, as required.

[0121] The process of FIG. 3 is preformed prior to commen-
cement of game play, However, a similar process is also
carried out during game play, when further passive data is
required from the game server 20.

[0122] In one embodiment it is not necessary for all of the
selected game passive data to be transferred before commen-
cement of game play. For example, only the passive data
which is required for the first three levels of the selected game
may be identified and transferred, using the process described
above with reference to FIG. 3, to the user’s device 30 before
game play commences.

[0123] During game play the passive data is downloaded to
the passive data storage device 302 at the user’s device 30 (in
the background). The passive data is then read from the pas-
sive data storage device 302 when required. In contrast the
active data is downloaded when required, and is not stored
permanently at the user’s device 30. The passive data storage
device 302 functions as a virtual drive at the user’s device 30
from which the passive data is read, in contrast to conven-
tional games which are read from a disc.

[0124] It may not be necessary for the process of FIG. 3 to
be carried out when the user selects the game for playing the
second and subsequent times, since the initial passive data
required to begin game play has already been downloaded to
the passive data storage device 302. However, in one embodi-
ment, an update check of the passive data storage device 302
may be undertaken each time the user connects to the game
server 20 to play a game, or at predetermined intervals. The
update check of the passive data storage device 302 makes a
determination as to whether any updated versions of the pas-
sive data stored in the passive data storage device 302 is
available at the server 20, and if so, downloads the updated
version. For example, if an updated version of the texture
glass becomes available it can be saved over the current
version of the texture grass stored in the passive data storage
device 302 and associated with the unique reference which
indicates the texture grass.

[0125] In addition, if the user has connected to the game
server 20 from a different user device 30, then the essential
minimum data may be required to be downloaded in accor-
dance with the process of FIG. 3. In another embodiment, if
the ping time or bandwidth of the user’s device 30 has altered
since the user was last connected to the server, then a check of
the passive data storage device 302 may be undertaken.

[0126] The quality of the updated files may be better (for
example in terms of resolution etc.) than the original files, yet
the overall game may still be smaller due to the amount of
redundant data deleted from the game. In this regard, the
visual and audio quality of a computer game may be
improved, enhancing the user’s experience whilst still reduc-
ing the overall size of the game.
Following download of the essential minimum data (including the game engine if required), the user can commence game play. As stated above, during game play the active data is downloaded when required. Consequently a constant two way link is required between the game server 20 and the user’s device 30 at all times during game play. In one embodiment the link is the internet using any of the standard internet protocols such as HTTP or FTP and the active game data is transferred over this link.

So that there are not any delays during game play whilst passive (and active) data is downloaded from the game server 20, it is necessary for the game server to monitor game play at the user’s device 30 and to determine a buffer zone which surrounds the player’s current position in the game. This buffer zone indicates all the passive (and active) data which may be required during the next predetermined period of game play. FIG. 4 illustrates schematically one example of a buffer zone.

As illustrated in FIG. 4, if the player is currently at the centre position (indicated by the solid black star) at time t=0, then the player could be anywhere within the first circle 1 at t=1, anywhere within the second circle 5 at t=5 and anywhere within the third circle 10 at t=10. In most games, the player is not required to move in one direction, such as direction A, B or C, and can turn around and double back on himself or can change direction, game play may not be linear and different users (or the same user) can play the same game in different ways, based on decisions taken during the game. Consequently, at any one time, the player could be at any one of a plurality of different positions each requiring different (or the same) active and passive data. For example, if the passive data buffer zone is defined by the dotted circle 10 in FIG. 4 and the active data buffer zone is defined by the dotted circle 1 in FIG. 4, then the game server 20 needs to determine all the passive data that may be required within the zones 0 to 10 and all the active data that may be required within the zones 0 to 1. The zones of the game may be defined in terms of time, distance, memory, rooms or levels within a game.

The active data buffer zone is smaller than the passive data buffer zone so as to avoid permanently storing (for any significant period of time/amount of data) active data at the user’s device 30 and thus prevents the game being played when connection to the server 20 is broken.

The kernel request module 318 monitors real time game play and tracks which direction the player is moving in and whether they are entering a new zone. Each zone within a game is linked to all the next possible zones which may precede/follow it. In one embodiment, the real time delivery module 312 and the recall manager 316 use the real time game play data from the kernel request module 318 to determine the active and passive data required respectively for the buffer zones surrounding the users current position. A request for the buffer zones data is transferred to the server 20 and as a result the required data is then transferred from the game server 20 to the user’s device 30 and stored in the passive data storage device 302 and the active data buffer 310 accordingly.

Then, for example, if the player travels in direction A, only the passive data within the dot/dash ellipse of FIG. 4, which is actually required is retrieved from the passive data buffer 322 during game play, and only the active data within the dot/dot ellipse of FIG. 4, which is actually required is retrieved from the active data buffer 310 during game play.

In one embodiment, the passive data predetermined period may be set as one level, in this case the game server 20 downloads each level of passive data one level before it is required.

In another embodiment, starting at z=0, the kernel request module 318 may determine in which direction the player is travelling, i.e. is the player moving in direction A, B or C. The real time delivery module 312 then determines what active data is required if the player were to move in the predicted direction, direction A. Thus, as illustrated in FIG. 4, the active data buffer zone comprises only the active data which is within ellipse A between z=0 and z=1, not the entire dotted circle 1. In addition, the recall manager 316 determines what passive data is required if the player were to move in the predicted direction, direction A. Thus, as illustrated in FIG. 4, the passive data buffer zone comprises the entire passive data which is required within ellipse A between z=0 and z=10, not the entire dotted circle 10.

In both instances, the buffer zones data constantly changes as a result of the actual direction in which the player travels, as illustrated in FIG. 5. As illustrated in FIG. 5 the player is initially at z=1 and the buffer zone for z=1 is indicated at z=1+1, z=1+5 and z=1+10. The player then moves in direction A to z=m and the buffer zone for z=m is indicated at z=m+1, z=m+5 and z=m+10. The player then moves again in direction A to z=n and the buffer zone for z=n is indicated at z=n+1, z=n+5 and z=n+10.

The buffer zone for the passive data is set to be larger than the buffer zone for the active data. This is because, in most instances, more passive data is required for one zone and passive data tends to have larger file sizes than the active data, thus the passive data requires longer download times. In addition, the active data is only stored temporarily, for a very short period of time, on the user’s device 30, at the active data buffer. The required active data is constantly being overwritten when it is no longer required. Therefore, a live connection between the user’s device 30 and the game server 20 is required during game play.

Since the active data is only stored temporarily on the users device 30 it is not possible for the user to play a downloaded game if they are not connected to the game server 20. In addition, it is not possible for a user to create a copy of the game (for resell for example), and the game cannot be used through a peer-to-peer sharing network, as the user’s device 30 does not store any more than a very limited period of active data required for game play. In addition, since the active data is only stored temporarily on the users device 30, the user’s device 30 is prevented from becoming slow over time as a result of accumulated active data which is no longer required.

According to one embodiment, the active data and passive data may be distributed by separate servers. For example, passive data can be distributed by a web server, since its delivery speed does not directly affect game play; a slower delivery speed simply means the user has to wait longer to begin playing the game. The delivery speed of active data, on the other hand, is more critical in the sense that the game cannot be played without it and most games will be prepared by the game conversion module 10 in a manner that requires a constant feed of active data in order for the game to be played. In this regard, therefore, it is more desirable to have a fast server for distributing the active data. According to one example, active data is distributed by a blade server and passive data is distributed by a web server.
As stated above the kernel request module 318 receives the current real time position of the user in the game during game play from the processor 34 and transfers this data to the real time delivery module 312 and the recall manager 316.

The real time delivery module 312 then determines what active data is required for the active data buffer zone surrounding the current position of the user in the game and transfers a request for the required active data to the active data manager 308. The indication of the required active data is retrieved by the real time delivery module 312 from the source code of the game for the zones surrounding the current position of the user in the game. The active data manager 308 requests the required active data from the game server 20 and when it receives the required active data, stores the required active data in the active data buffer 310. Consequently, the active data buffer 310 contains active data which is required for the active data buffer zone surrounding the current position of the user in the game.

The recall manager 316 determines what passive data is required for the passive data buffer zone surrounding the current position of the user in the game. The recall manager 316 then transfers an indication of the required passive data to the passive data manager 304. The indication of the required passive data is retrieved by the recall manager 316 from the source code of the game for the zones surrounding the current position of the user in the game. The passive data manager 304 transfers indications to the passive data storage device manager 306 and the passive data storage device manager 306 determines if any of the required passive data is not stored in the passive data storage device 302. The passive data storage device manager 306 then transfers the references associated with the required passive data which is not stored in the passive data storage device 302 to the passive data manager 304.

The passive data manager 304 requests the required passive data which is not stored in the passive data storage device 302 from the game server 20. The game server 20 then transfers the required passive data to the passive data storage device manager 306 via the passive data manager 304, and the passive data storage device manager 306 stores the required passive data in the passive data storage device 302.

In another embodiment, the kernel request module 318 receives the current real time position of the user in the game during game play from the processor 34 and transfers this data to the game server 20. The game server 20 also determines what active data is required for the active data buffer zone surrounding the current position of the user in the game and transfers the required active data to the active data manager 308. The game server 20 also determines what passive data is required for the passive data buffer zone surrounding the current position of the user in the game and transfers a list of references for the required passive data to the passive data manager 304. The passive data manager 304 determines what passive data from the list of passive data is not stored in the passive data storage device 302 and transfers a request for the passive data not stored at the user’s device 30 to the game server 20. The game server 20 then transfers the requested passive data to the passive data storage device manager 306 via the passive data manager 304, and the passive data storage device manager 306 stores the required passive data in the passive data storage device 302.

The passive data which is required for the passive data buffer zone is then transferred from the passive data storage device 302 to the passive data buffer 322. Consequently, the passive data buffer 322 temporarily stores the passive data which is predicted to be required within the next predetermined period of game play. This increases the speed of game play since the passive data which is likely to be required is kept in a smaller storage device (the passive data buffer 322) and thus can be identified and retrieved quicker then if the passive data was retrieved directly from the passive data storage device 302 when required.

The passive data buffer 322 temporarily stores a small subset of the passive data stored in the passive data storage device 302. As stated above, the passive data storage device 302 stores all the passive data required for all the games played by the user. Thus, the passive data storage device 302 can store a large amount of passive data depending on the number of different games played by the user. As with the active data buffer 310, the data stored in the passive data buffer 322 is constantly being written over with new required passive data determined based on the new current position of the user in the game.

The passive data which is stored in the passive data buffer 322 is fed into the game constructor module 320, in sequential order. For example, all the passive data which is required at zone z=n, is fed into the game constructor module 320 from the passive data buffer 322, before all the passive data which is required at zone z=n-1.

The real time delivery module 312 retrieves the required active data from the active data buffer 310. The game synchroniser module 314 then synchronises the transfer of the active data from the real time delivery module 312, and the passive data from the game constructor module 320 to the kernel request module 318, such that for example, the active data which is required at z=5 is received at kernel request module 318 with the passive data which is required at z=5.

The kernel request module 318 then combines the synchronised active and passive data and transfers it to the processor 34.

The active data which is required for the active data buffer zone is temporarily stored in the active data buffer 310, and the passive data which is required for the passive data buffer zone is temporarily stored in the passive data buffer 322. However, only some of the active data and passive data is actually required, depending on the user’s interaction with the game and a lot of the active and passive buffer zone data is not actually required during game play.

The process describe above is constantly performed during game play, since the kernel request module 318 is constantly receiving an indication of the user’s current (real time) position in the game from the processor 34. This data is constantly being transferred to the real time delivery module 312 and the recall manager 316 such that the active data buffer zone and passive data buffer zone can be updated. If the connection to the server 20 is broken then the user will only be able to continue playing until the active data in the active data buffer 310 has been used. Then game play will not be able to continue, as a result of the lack of active data until the connection to the server 20 is re-established.

It is also possible for the game user apparatus 300 to be launched through a conventional internet browser. For example by clicking a game listed on a website hosted by the game server 20, an extension of the browser automatically launches the game user apparatus on the user’s device 30 and
begins to initiate the essential minimum download or game play (i.e. active and passive data stream) if the game has been played before.

[0151] The invention has been described with particular illustrative embodiments. It is to be understood that the invention is not limited to the above-described embodiments and that various changes and modifications may be made by those of ordinary skill in the art without departing from the scope of the invention.

1. A user game apparatus configured to facilitate game playing according to which first and second separate supplies of game data are received from a remote game distribution server and combined during game play responsive to game playing events, the apparatus comprising:
   a request module arranged to receive a first supply of data from a first data path and a second supply of data from a second data path and operable to combine said first and second data to produce a natural game play experience; and
   a game synchroniser configured to control timing of supply of the first supply of data and the second supply of data from respective data buffers.

2. The user game apparatus according to claim 1, wherein the first data has an average file size smaller than an average file size of the second data.

3. A user game apparatus configured to facilitate game playing according to which active game data and passive game data are received from a remote game distribution server and combined during game play responsive to game playing events, the apparatus comprising:
   a first module arranged to receive a supply of active data from an active data path and a supply of passive data from a passive data path, said first module being operable to combine said active and passive data to produce a natural game play experience; and
   a game synchroniser configured to control timing of the supply of the active data from the active data path to the first module and the supply of the passive data from the passive data path to the first module.

4. The user game apparatus according to claim 3, further comprising:
   a real-time delivery module supplying active data from an active data buffer responsive to the control of the game synchroniser.

5. The user game apparatus according to claim 4, wherein loading of the active data buffer is controlled by an active data manager communicating with the remote game distribution server.

6. The user game apparatus according to claim 5, wherein the first module is configured to supply game play event indicators along the active data path.

7. The user game apparatus according to claim 6, wherein game play event indicators are supplied via the active data path to a remote game distribution server, and wherein the active data manager receives from the remote game distribution server active data required for future game play.

8. The user game apparatus according to claim 6, wherein the real-time delivery module is configured to process the game play event indicators to determine what active data is required for future game play, and the active data required for future game play is requested from the remote game distribution server.

9. The user game apparatus according to claim 7 or 8, wherein the active data required for future game play is staged into the active data buffer ahead of being required by the first module.

10. The user game apparatus according to any one of claims 7 to 9, wherein the active data required in future game play is staged into the active data buffer in dependence upon the game play event indicators dictated by the game code.

11. The user game apparatus according to any one of claims 4 to 10, wherein the active data stored in the active data buffer is deleted once used.

12. The user game apparatus according to any one of claims 4 to 11, wherein the active data stored in the active data buffer is deleted after a predetermined period of time; is deleted after the active data buffer capacity is exceeded; and/or is deleted after implementation in game play.

13. The user game apparatus according to any one of claims 3 to 12, further comprising:
   a game constructor module operable construct and run a clone version of the game that is substantially complete save for the absence of active data.

14. The user game apparatus according to any one of claims 3 to 13, wherein the first module is configured to supply game play event indicators along the passive data path.

15. The user game apparatus according to claim 14, wherein the game play event indicators are supplied to the remote game distribution server, and wherein a passive data manager receives from the remote game distribution server an indication of what passive data is required for future game play.

16. The user game apparatus according to claim 14, wherein a recall manager is configured to process the game play event indicators to determine what passive data is required for future game play, and to transfer to a passive data manager an indication of what passive data is required for future game play.

17. The user game apparatus according to claim 15 or 16, wherein the passive data manager is configured to determine if the passive data required for future game play is available in a passive data store.

18. The user game apparatus according to claim 17, wherein the passive data manager is operable to request the passive data required for future game play from the remote game distribution server and to store the requested passive data in the passive data store in the event it is not already available in the passive data store.

19. The user game apparatus according to any one of claims 3 to 14, wherein the passive data comprises a plurality of passive data objects, each passive data object associated with a unique reference.

20. The user game apparatus according to any one of claims 15 to 17, wherein the passive data comprises a plurality of passive data objects, each passive data object associated with a unique reference, and wherein the indication of what passive data is required for future game play comprises a plurality of unique references.

21. The user game apparatus according to claim 18, wherein the passive data comprises a plurality of passive data objects, each passive data object associated with a unique reference, and wherein the indication of what passive data is required for future game play comprises a plurality of unique references and the request for passive data from the remote game distribution server comprises a plurality of unique references.
22. The user game apparatus according to any one of claims 18 to 21, wherein the passive data path further comprises a passive data path from which passive data may be transferred to the game constructor module.

23. The user game apparatus according to claim 22, wherein the passive data is supplied from the passive data store to said game constructor module via the passive data buffer.

24. The user game apparatus according to claim 23 or 24, wherein the passive data required in future game play is staged into the passive data buffer in dependence upon game play events as dictated by the game code.

25. The user game apparatus according to any one of claims 4 to 24, wherein an amount of active data stored in the active data buffer is less than an amount of passive data stored in the passive data buffer.

26. The user game apparatus according to any one of claims 3 to 25, wherein the active data comprises a plurality of active data objects with file sizes that are on average smaller than the average file size of passive data objects.

27. A method of operating user game apparatus provided at a user terminal and configured to facilitate game playing, the method comprising:
   receiving a supply of active data from a remote game distribution server and a supply of passive data from a remote game distribution server;
   controlling timing of the supply of the active data and the supply of the passive data; and
   combining said active and passive data during game play responsive to game playing events to produce a natural game play experience.

28. The method according to claim 27, further comprising:
   supplying game play event indicators to the remote game distribution server; and
   receiving from the remote game distribution server active data required for future game play.

29. The method according to claim 27, further comprising:
   supplying game play event indicators to the real-time delivery module;
   processing the game play event indicators to determine what active data is required for future game play; and
   requesting the active data required for future game play from the remote game distribution server.

30. The method according to claim 28 or 29, further comprising:
   supplying the active data required for future game play into an active data buffer.

31. The method according to claim 30, further comprising:
   deleting the active data from the active data buffer once used.

32. The method according to claim 30, further comprising:
   deleting the active data from the active data buffer after a predetermined period of time; after the active data buffer capacity is exceeded; and/or after implementation in game play.

33. The method according to any one of claims 27 to 32, further comprising:
   constructing and running a clone version of the game that is substantially complete save for the absence of active data.

34. The method according to any one of claims 27 to 32, further comprising:
   supplying game play event indicators to the remote game distribution server; and
   receiving from the remote game distribution server an indication of what passive data is required for future game play.

35. The method according to any one of claims 27 to 32, further comprising:
   supplying game play event indicators along the passive data path to a recall manager; and
   processing the game play event indicators to determine what passive data is required for future game play.

36. The method according to claim 34 or 35, further comprising:
   determining if the passive data required for future game play is available in a passive data store.

37. The method according to claim 36, further comprising:
   requesting passive data from the remote game distribution server in the event it is not already available in the passive data store; and
   storing the requested passive data in the passive data store.

38. A computer program product comprising programme code means for performing the method according to any one of claims 27 to 37.

39. A computer readable medium recorded with computer readable code arranged to cause a computer to perform the method according to any one of claims 27 to 37.

40. A computer programme code means for performing the method according to any one of claims 27 to 37.