(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 November 2006 (30.11.2006)

PCT

(10) International Publication Number
WO 2006/127596 A2

(51) International Patent Classification:
H01M 2/04 (2006.01)

(21) International Application Number:
PCT/US2006/019731

(22) International Filing Date: 22 May 2006 (22.05.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/683,099 20 May 2005 (20.05.2005) US

(71) Applicant *(for all designated States except US)*: **HILLCREST LABORATORIES, INC.** [US/US]; Suite 450, 15245 Shady Grove Road, Rockville, MD 20850 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **STANCHFIELD, Scott** [US/US]; 20405 Apple Harvest Circle, Apt. B, Germantown, MD 20876 (US).

(74) Agent: **KALIDINDI, Krishna, V.**; POTOMAC PATENT GROUP PLLC, P.O.Box 270, Fredericksburg, VA 22404 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 2006/127596 A2

(54) Title: DYNAMIC HYPERLINKING APPROACH

(57) Abstract: A method of returning target scenes from a user link request is disclosed. The method comprises the steps of: receiving a user link request on a user interface; comparing the received user link request to a plurality of entries in a lookup table; for each entry in the plurality of entries in the lookup table that matches the received user link request, identifying a target scene that corresponds to the matched entry in the lookup table; determining a closest one of the target scenes if multiple entries in the lookup table match the user link request; and transitioning to the closest target scene.

DYNAMIC HYPERLINKING APPROACH

BACKGROUND

The present invention relates, generally, to user interfaces and methods associated therewith and, more specifically, to navigation between user interface icons organized in a hierarchical manner.

User interfaces are ubiquitous in today's society. Computers, cell phones, fax machines and televisions, to name a few products, all employ user interfaces. User interfaces are intended to provide a mechanism for users to easily access and manipulate the, sometimes complex, functionality of the devices that they support. An example of a user interface is found in U.S. Patent Application Serial No. 10/768,432, filed on January 30, 2004, entitled "A Control Framework with a Zoomable Graphical User Interface for Organizing, Selecting and Launching Media Items", the disclosure of which is incorporated here by reference. Typically, such user interfaces employ remote, handheld devices to provide inputs to their respective applications.

User interfaces facilitate navigation within an increasingly complex array of choices in a home entertainment system for example. Objects or icons within a user interface may be organized logically in a hierarchical manner. A user may select an icon on a home entertainment system user interface to indicate his or her desire to watch TV, play computer games, access the internet, etc. via a monitor/display for example. Once the choice for watching TV is indicated, the user may be presented with icons (for selection) that represent broadcast channels, premium channels, sports channels, etc. Each of these icons may be actuated to provide additional levels of choices. For example, under broadcast channels, the network channels may be presented.

As the choices and levels of organization of the interface objects/icons increase, it would be desirable to transition between user choices in a rapid manner.

SUMMARY OF THE INVENTION

In one embodiment, a method of returning target scenes from a user link request is disclosed. The method comprises the steps of: receiving a user link request on a user interface; comparing the received user link request to a plurality of entries in a lookup table; for each entry in the plurality of entries in the lookup table that matches the received user link request, identifying a target scene that corresponds to the matched entry in the lookup table; determining a closest one of the target scenes if multiple entries in the lookup table match the user link request; and transitioning to the closest target scene.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings illustrate exemplary embodiments of the present invention, wherein:

FIG. 1 depicts a user interface according to an exemplary embodiment of the present invention;

FIGS. 2 - 4 depict exemplary hierarchical levels of objects/icons in a user interface;

FIGS. 5A and 5B depict exemplary hierarchical level of objects/icons corresponding to the objects of FIG. 4;

FIG. 6-7B illustrate exemplary free-space pointing devices;

FIGS. 8A and 8B illustrate exemplary arrangement of icons on a user

interface in accordance with an exemplary embodiment of the present invention;

FIGS. 9A and 9B illustrate exemplary display of related information for a particular program on a user interface in an exemplary embodiment of the present invention;

FIGS. 10-11B illustrate an exemplary hierarchical tree/node structure for programming information in an entertainment system in exemplary embodiment of the present invention;

FIGS. 12A-12F illustrate an exemplary transitioning sequence on a user interface between scenes in accordance with an exemplary embodiment of the present invention;

FIG. 13 illustrates an exemplary system of the present invention; and

FIG. 14 illustrates an exemplary method of the present invention.

DETAILED DESCRIPTION

The following detailed description of the invention refers to the accompanying drawings. The same reference numbers in different drawings identify the same or similar elements. Also, the following detailed description does not limit the invention. Instead, the scope of the invention is defined by the appended claims.

Exemplary embodiments of the present invention provide a method for navigating between, and returning targets for, user selections on a user interface. More specifically, a "closest" target (from potentially multiple targets) is returned as described in more detail herein below. User interfaces as they pertain to exemplary embodiments are described first and specific exemplary methods for returning targets are then described.

An exemplary user interface (UI) 100 is illustrated in FIG. 1. UI 100 may include a plurality of icons 110-160 each representing a choice for user selection. The user may highlight and depress (or "click") one of these icons such as icon 110 to indicate his or her preference for watching TV. A user input device may be utilized to perform this function. UI 100 may function as the home or root user interface.

An exemplary input device (or free-space pointing device) may resemble a mouse such as a wireless mouse 600 depicted in FIG. 6. Button presses on mouse 600 correspond to actuation of buttons 602 and 604 and scroll wheel 606 for example.

Another exemplary free-space pointing device 700 may resemble a loop-shaped device such as that illustrated in FIGS. 7A and 7B and designed by HillCrest Labs of Rockville, MD. Free-space pointing device 700 includes buttons 702 and 704 and scroll wheel 706 (corresponding to buttons 602 and 604 and scroll wheel 606 of FIG. 6).

Referring back to FIG. 1, actuation of icon 110 may result in displaying UI 200 of FIG. 2. UI 200 may present choices under the "watch TV" category of FIG. 1. The choices may include icons 210-230 representing premium channels, broadcast channels and sports channels for example. UI 200 may also include navigational icons 250-254. Clicking on home icon 250 may result in return to UI 100 (or, "home") of FIG. 1 for example.

If a user chooses to watch premium channels, then actuation of icon 210 may result in displaying UI 300 of FIG. 3 which may include icons 310-320 corresponding to exemplary premium channels and navigational icons 350-354.

Actuation of icon 310 on UI 300 may result in displaying UI 400 of FIG. 4

which may include icons 410-420 corresponding to exemplary genres and navigational icons 450-454. Actuation of icons 410 and 420 may result in displaying UI 500 of FIGS. 5A and 5B respectively which may include icons 510, 520, 530 and 540 corresponding to exemplary movie titles and navigational icons 550-554. The movie titles are also exemplary and in some cases, such as "The Matrix Reloaded Preview" and "The Matrix - Behind Scenes", may be fictitious but may be based on actual movie titles and have been included for purely illustrative purposes.

While each of the icons are depicted as having an oval shape, they are not restricted to this shape. The icons may be rectangular, square, circle or any other shapes. The icons may also be arranged in multiple rows as illustrated in user interface 800 FIG. 8A (in a rectangular shape) which also includes navigational icons 850, 852 and 854.

For each movie title (FIGS. 5A, 5B or 8A), the user may wish to view the movie; the user may also be presented with a description of the movie and/or a list of related title as illustrated on user interface 900 in FIG. 9 that also includes navigational icons 950, 952 and 954.

The user interface illustrated may be referred to as a zoomable user interface (ZUI) since highlighting one of a plurality of icons magnifies the highlighted icon in exemplary embodiments as illustrated in user interface 800 of FIG. 8B for example which also includes navigational icons 850, 852 and 854. The '432 application describes the zoomable aspect.

An overall exemplary hierarchical structure for the icons of FIGS. 1-5B is illustrated as nodes in a tree in FIG. 10. For the purposes of this invention, each node may be referred to as a scene. The illustration of FIG. 10 is purely illustrative

and not exhaustive. In implementation, several additional networks may be available (i.e. besides HBO® and SHOWTIME®); for each network, additional genres may be available and for each genre, additional movie titles (besides the two that are illustrated) may be available.

The language definition for each scene is represented as a scalable vector graphic (SVG). SVGs are used as they describe shapes on a display page, movements on a display page and location of graphics as they appear on a display page within a graphical display.

Programming information for a particular channel/network may be received by the server from the respective network on a periodic basis. Such programming may be dynamic in nature. That is, a movie title under the action genre for HBO for one period (a week or a month) may differ from a movie title under the action genre for HBO for a second period.

Each of the nodes/scenes may have a unique description or definition that may be stored in a server. The scene definition therefore may rely on information received from the networks. The scene representing "The Terminator" in FIG. 10 may have the exemplary definition: *home\SHOWTIME\movies\sci-fi\The Terminator*.

The scene representing "The Matrix" may have the following two exemplary definitions since The Matrix occurs twice: *home\SHOWTIME\movies\sci-fi\The Matrix* and *home\HBO\ movies\action\The Matrix*.

The definition in this exemplary embodiment may be in a folder/subfolder type of hierarchy. Other formats may also be used to define the scenes. The definition for a scene may be viewed as the path from the home location

to the scene location. The server may store a description (or definition) for all scenes that are currently available.

In the example described above, a plurality of generic identifiers may be assigned under the genre. The generic identifiers may be termed as *movie-0, movie-1, . . . movie-n*. The generic identifiers may have different movies associated with them based on programming for a particular time period. For the month of November 2005 for example, *movie-0* may correspond to "The Matrix" and *movie-1* may correspond to "Mission: Impossible" In April 2006, *movie-0* may correspond to "Spiderman" and *movie-1* may correspond to "Mission: Impossible II".

Each of the program titles (such as "Mission: Impossible" and "The Matrix") may have unique identification associated therewith such as a numeric or an alphanumeric code for example. Such identification may be included in metadata corresponding to the movie and may be received or known by the server.

A user highlighting "The Matrix" under *HBO\Movies\Action*, for example, may also be presented with related titles. The related titles/movies may be those having similar story lines, genre or those having common actors, etc. The related titles/movies may also be included in the metadata received by the server.

A user wishing to obtain information (such as a preview clip or description) or to view/play a related movie may click on an icon representing the related movie (movie 21, movie 36, etc. of FIG. 9). This may serve as a user link request. The request for information on and for viewing of the related movie (movie 21 for example) may be represented on the movie 21 icon for example.

For each movie title or program available on the server, an entry may be created in a lookup table based on an identification generated by the server

representing a user link request. Separate entries may be created for each aspect of a particular title. The separate aspects may be movie information (such as preview) and movie play for example. For each of these entries, a location (i.e. path of a scene described above) where the desired information such as movie information or movie play (a "target scene") may be found may be listed in the corresponding column.

Because a particular user link request may have multiple resulting scenes, the link request may be listed multiple times in the table - each having a different corresponding resulting scene or path.

During runtime (i.e. while the user is navigating the user interface), a link request received from a user may be searched for in the table. If the received link request matches an entry in the table, the corresponding location information is obtained. If multiple entries match the received link request, then the information for multiple corresponding locations (each corresponding to one of the multiple entries) is obtained and the closest these corresponding locations selected using an algorithm described below.

During runtime, using the exemplary setup illustrated in FIG. 10, if a user seeks information on related movies from "The Matrix" under *Home\HBO\Movies\Action* for example, the server may provide the two choices in the example illustrated in FIG. 10 ("The Matrix Reloaded Preview" and "The Matrix - Behind Scenes").

For "The Matrix Reloaded Preview", there are two potential resulting scenes. The scenes in this example have the paths *Home\The Matrix Reloaded Preview* and *Home\SHOWTIME\The Matrix Reloaded Preview*. A scene representing

"The Matrix - Behind Scenes" may be *Home\HBO\Documentaries\The Matrix - Behind Scenes* or *Home\SHOWTIME\Movies\The Matrix - Behind Scenes*.

If the user chooses "The Matrix Reloaded Preview" for example, the server computes the shortest distance to "The Matrix Reloaded Preview" from the current scene.

In determining the shortest distance, the server attempts to minimize the number of steps or levels that have to be traversed up (i.e. go up as little as possible). This may be accomplished by picking the target having the longest common prefix (as it relates to the scene definition or path) with the current scene.

Then, it attempts to minimize the number of steps or levels that have to be traversed down (i.e. go down as little as possible). For the "going down" part, the server examines paths to each possible target and selects the path having the fewest "\" in exemplary embodiments.

In the example above, if the chosen scene is "The Matrix Reloaded Preview", the current location/scene is *Home\HBO\Movies\Action\The Matrix*. The two possible destination scenes may be *Home\SHOWTIME\The Matrix Reloaded Preview* and *Home\The Matrix Reloaded Preview*. The common prefix for the current scene and the first destination scene is Home. The common prefix for the current scene and the second destination scene is also Home.

Both destination scenes in this example have the same length common prefixes with respect to the current scene. This implies that the same number of levels have to be traversed up regardless of either destination scenes. However, the number of levels to traverse down from Home varies between the destination scenes. For the first destination scene, two levels have to be traversed. For the second destination

scene, only one level has to be traversed down. In this case, the server chooses the second destination (i.e. *Home\The Matrix Reloaded Preview*).

If the user chooses "The Matrix - Behind Scenes", then the two destination scenes may be *Home\HBO\Documentaries\The Matrix - Behind Scenes* or *Home\SHOWTIME\Movies\The Matrix - Behind Scenes*. In this scenario the current scene (i.e. *Home\HBO\Movies\Action\The Matrix*) has a longer common prefix with *Home\HBO\Documentaries\The Matrix - Behind Scenes* than with *Home\SHOWTIME\Movies\The Matrix - Behind Scenes*. The server then chooses the first destination scene.

If only one destination scene is available, then the shortest distance need not be determined and the one destination scene may be selected. As illustrated in FIG. 11A, from "The Matrix", a user's desire for background information results in only one scene corresponding to "The Matrix - Behind Scenes". This destination may then be selected.

If the distance to multiple destinations is identical, then any one of the destinations may be selected. As illustrated in FIG. 11B, from "The Matrix", a user's desire for background information results in two scenes corresponding to "The Matrix - Behind Scenes" (*HBO\Documentaries\The Matrix - Behind Scenes* and *HBO\Events\The Matrix - Behind Scenes*). In this scenario, either of the destination scenes may be selected.

The transition from the source scene to the destination scene may be visually presented on the user interface utilizing the zoomable aspect of the user interface. That is, for example, the transition from *Home\HBO\Movies\Action\The Matrix* to *Home\HBO\Documentaries\The Matrix - Behind Scenes* in FIG. 10 may be

displayed as a series of transitioning sequence interfaces 1200 illustrated in FIGS. 12A-12F.

At each stage of the transitioning sequence, the scene of interest (1210, 1220, 1230, 1240, 1253 and 1260) or relevance may be displayed more prominently on the user interface 1200 than other scenes (1215, 1225, 1235, 1245, 1247 and 1255) at that hierarchical level (illustrated in FIG. 10). The navigational icons 1250, 1252 and 1254 appears in the user interface as well.

Navigational icons illustrated in each of FIGS. 2-5B facilitate navigation to home (icon 550 for example), up one level (icon 554) or to previous scene (icon 552).

The methods described herein may be implemented using known Java techniques. The server may be remotely located and user interaction via a handheld device may be facilitated over a network connecting a receiver (receiving user actuation from the handheld device) to the server. An exemplary system 1300 is illustrated in FIG. 13. System 1300 includes a server 1310, a network 1320, a receiver 1330 and a handheld device 1340. Communication between the handheld device 1340 and the receiver 1330 may be over wireless medium such as Bluetooth for example.

Exemplary methods described above may be illustrated with reference to FIG. 14. A user link request may be received at 1410. The received request is compared to a plurality of entries in a lookup table at 1420. For each entry in the plurality of entries in the lookup table that matches the received user link request, a target scene that corresponds to the matched entry in the lookup table is identified at 1430. A closest one of the target scenes is determined if multiple entries in the

lookup table match the user link request at 1440. The closest target scene is transitioned to at 1450.

The above-described exemplary embodiments are intended to be illustrative in all respects, rather than restrictive, of the present invention. Thus the present invention is capable of many variations in detailed implementation that can be derived from the description contained herein by a person skilled in the art. For example, while the description focused on navigating between programs in multiple locations in a home entertainment system, exemplary embodiments may be equally applicable for determining a closest branch of a retail chain having multiple locations that is within a same jurisdiction for ensuring a particular sales tax rate.

All such variations and modifications are considered to be within the scope and spirit of the present invention as defined by the following claims. No element, act, or instruction used in the description of the present application should be construed as critical or essential to the invention unless explicitly described as such.

Text of U.S. Provisional Application No. 60/683,099 entitled "Dynamic Hyperlinking Approach" is included on the following pages:

Episode I: The Static Menace

Our first attempt at a hyperlinking approach has uncovered several problems:

- Hyperlinks are only resolvable statically. We cannot create hyperlinks from the search page or from pages that set variables via JavaScript based on user interaction. This allows invalid links to be displayed, with no choice for the client other than to "do nothing" or pre-resolve all links via server calls (not a good idea for performance...).
- Hyperlinks can only be generated for a single root SVG. This makes debugging during development burdensome, as developers either need to start all sessions from the home SVG or regenerate the link resolution file for the page they want to start on.
- Hyperlinks can only be resolved under the selected root SVG. If a developer chooses different root, they cannot follow links outside the subtree rooted at that SVG. If a server allowed multiple roots based on different clients, we would have to clone the database for each root SVG.
- Cycles in SVGs cannot be detected, which could cause the link resolution generation to endlessly loop. We would need to artificially restrict the depth of resolution, which could cause links to stop responding if a user navigates through long chains/cycles.
- There is no way to determine if a link is bad until we try to resolve it, and then there is nothing for the client to do but "nothing". This makes it appear that a hyperlink is broken, when in fact it just wasn't something valid to link to. We need some way of creating a new page when this happens to display whatever data
- Whenever any SVGs change, the link pre-resolution must be re-run for all SVGs.

Episode IV: A New Hope

Instead of pre-instantiating the entire scene instance graph, let's take a look at the data from a new angle. In fact, let's reverse the data a bit...

Think about the following artifacts:

- K1: Knowledge of which queries can return any specified entity.
- K2: Knowledge of which SVGs link to which SVGs (via scene transitions).
- K3: Knowledge of which SVGs run which queries.
- K4: Knowledge of which SVGs pass which data to their target scenes, and the intended usage of that data. (This is usually represented as something like sim_movies[4], but we'll keep this just as knowledge that "some" result from a similar movie query was passed down. The position of it doesn't really matter, so we only need one target relation regardless of how many scenes are in a given shelf.)
- K5: Derive from K3+K4 which query results can be displayed on which pages for which usage keywords.

When we want to resolve a link, we know the following information:

- Where in the scene instance graph we are
- The target entity
- The intended usage keywords

For example, a link in a page might look like the following:

```
<a id="sim_link_2"
zui:metadata="${linkTo(${sim_movies[2]},movieDetail)}">
```

The linkTo function takes one or more parameters. The first parameter is the target entity, and the remaining parameters are intended usage keywords.

For a target page to be considered, it must include all of the intended usage keywords. The target page itself does *not* specify the keywords; the page that links to it does. For example, a shelf might include:

```
<zui:scene … xlink:href="detail.svg">
  <zui:variable name="this" usage="movieDetail"
value="${movies[7]}"/>
</zui:scene>
```

Any number of usage keywords can be specified, separated by commas. This example indicates that this particular usage of "detail.svg" as a subscene is for displaying movieDetail.

Now the fun part…

Assuming we have the knowledge described above, we can start a very simple algorithm:

1. Determine from K1 which queries can return the requested entity.
2. Determine from K5 which pages are possible targets based on that query and the intended usage keywords.
3. Determine from K2 the shortest path to one of those pages. (See "shortest path" definition below")

All of the knowledge logically belongs to the server. I say *logically*, because the *physical* location can be changed based on how the server proxy is managed. If we determine that the data makes more sense to be physically located on the server, the client-side server proxy will forward the algorithmic request to the server. If we determine that some of the data (perhaps the SVG linkage and query usage information) makes sense to be physically located on the client, the client-side server proxy would only ask the server for which queries can return a given entity.

Episode V: The Missing Link Strikes Back

There are two possible "missing link" scenarios:

1. No pages in the scene instance graph transition to sub-scenes that would display anything from a query that would return the target entity.
2. Some page runs a query but the result wouldn't normally appear on a transitioned page. (For example, we run a query that sets "movies", and the scene transitions only use movies[0] through movies[5], but the entity we want is returned as movies[6])

Let's resolve the second scenario first, as it should be easy. Just pretend that the querying page *had* the missing entity and display the new sub-scene. Poof! This would allow transition to an entity that normally wouldn't have been displayed, *and* its parent contains related movies. Once we fix the "up arrow", we could go from "BioDome" (which no one would *ever* want displayed in a brick) up to a "Comedies" brick, and move to other worthwhile movies.

The first scenario is a bit trickier... There is no existing place in the scene instance graph that could display the entity for the requested keywords. We can approach this from a few angles:

1. Check to see if we can resolve *without* the usage keywords. If so, use the target. For example, suppose the usage keywords are "starz,movieDetail" and no page under starz would display the result. If some page "hbo,movieDetail" could display it, ignoring the keywords would allow us to link to HBO to see the entity.
    a. Do we ignore all keywords, or successively remove them? (Highest priority comes first, as in "movieDetail,starz", which we reduce to "movieDetail" then no keywords.
    b. Should this be an option?
2. Create a new sub-scene of the current page. The drawback here is that the new page may seem "out of place".

3. Create "dummy" subscenes under other pages (like search) that take no parameters but are not "linkable" from the other pages. This may require a front-end change to make it invisible.

For now we have selected option 3. We created dummy scenes for each type of object.

Episode VI: Return of the Shortcuts

During implementation, several shortcuts were discovered that make this process significantly easier. The following is a discussion of how the actual algorithm works at build time and at runtime.

Build Time – Preparation

Before we run the ZuiBrowser, we must prepare two sets of data:

1. The set of query/position pairs for each entity ("reverse query lookup map")
2. The set of potential links ("potential targets map")

This currently assumes all entities in the universe are known so they can be pre-processed.

Preparation of the potential links is simple. During the software build process, all queries are executed and the position of the resulting entities is noted. This data is stored in a database or as a serialized hash table of query/result list pairs.

The potential links are slightly trickier until two simplifications are discovered:

1. Data flow analysis can be simplified by walking all SVGs and executing variable assignments using query types as values (instead of query results).
2. Closest node resolution can be determined with a simple check for the longest prefix of the path strings that represent the current and potential target nodes.

Data Flow Analysis

We care about where query results are passed. This can be determined using the following approach:

1. Anywhere we see a <zui:query> tag, *do not run the query*; instead, set the result variable value to "[query:*queryName*]".
2. Create a new variable expression parser behavior that does the following:
   - de-index a variable (e.g. ${movies[0]}) simply removes "[" and "]" surrounding a variable's value
   - de-referencing (e.g. ${movie.actor}) is currently ignored and reports a message (later we can add supporting metadata to resolve these into result types if desired)
3. Walk the SVGs and generate a tree of their relationships. We only track scene and brick inclusions with variables/query results passed to them. Each variable assignment is tracked with its name, expression and usage keywords.
4. Walk the resulting tree executing each variable assignment using the behavior from #2.

The result of this process is that we end up with some variables having values like query:*queryName*. Any such variables in a scene definition indicate a potential link target. For example, **query:Hillcrest_Action** with keyword **movieDetail** indicates a potential target for action movies.

All targets are stored in a simple text file with the following format:

```
U usageKeywordString
Q queryName
n=scenepath
n=scenepath
n=scenepath
n=scenepath
Q queryName
```

```
n=scenepath
n=scenepath
n=scenepath
n=scenepath
U usageKeywordString
Q queryName
n=scenepath
n=scenepath
n=scenepath
n=scenepath
n=scenepath
Q queryName
n=scenepath
n=scenepath
n=scenepath
n=scenepath
...
```

For example:

```
U movieDetail
Q Hillcrest_Action
0=zuipath:/trans_14/watchTV/trans_0/trans_4!movies_2/detail_0
0=zuipath:/trans_14/watchTV/trans_1/trans_4!movies_2/detail_0
0=zuipath:/trans_14/watchTV/trans_6/trans_4!movies_2/detail_0
0=zuipath:/trans_14/watchTV/trans_5/trans_4!movies_2/detail_0
0=zuipath:/trans_14/watchTV/trans_4/trans_4!movies_2/detail_0
0=zuipath:/trans_14/watchTV/trans_2/trans_4!movies_2/detail_0
0=zuipath:/trans_14/watchTV/trans_3/trans_4!movies_2/detail_0
0=zuipath:/trans_14/watchTV/trans_8/trans_4!movies_2/detail_0
0=zuipath:/trans_14/watchTV/trans_9/trans_4!movies_2/detail_0
1=zuipath:/trans_14/watchTV/trans_0/trans_4!movies_2/detail_1
1=zuipath:/trans_14/watchTV/trans_1/trans_4!movies_2/detail_1
1=zuipath:/trans_14/watchTV/trans_6/trans_4!movies_2/detail_1
...
```

This format can easily be read into a three-keyed hash table (keyed by usage keyword string, query name, position). Each value in the hash table is a list of possible target

pages for a usage/query/position.

Once we both sets of data, we can move to the runtime component of the link resolution.

Runtime – Link Resolution

To resolve the links, we use the build-time data in the following algorithm. For example purposes, suppose the link were specified as **${linkTo(${movie},movieDetail)}**.

1. Resolve the entity and keywords in the linkTo expression. This yields query name/position pairs.
2. Use the reverse query lookup map to determine which potential target scenes display a result from that query at that position. This results in a list of potential targets.
3. Narrow the list of potential targets to those with the closest common parent to the current page position. The key realization here is that we need no tree walking; all path specifications have parent information encoded in them. This step becomes simply comparing strings and keeping the potential targets that have the longest common prefix with the current page. The result of this step is one or more potential pages.
4. Further narrow the list to the potential targets with the shortest overall path. This can be easily determined by counting "/" and "!" symbols in the path string. This results in one or more potential target pages.
5. All remaining targets are equally close; return the first target in the remaining targets list.

Thus, link resolution becomes very fast and efficient, as well as simple to determine.

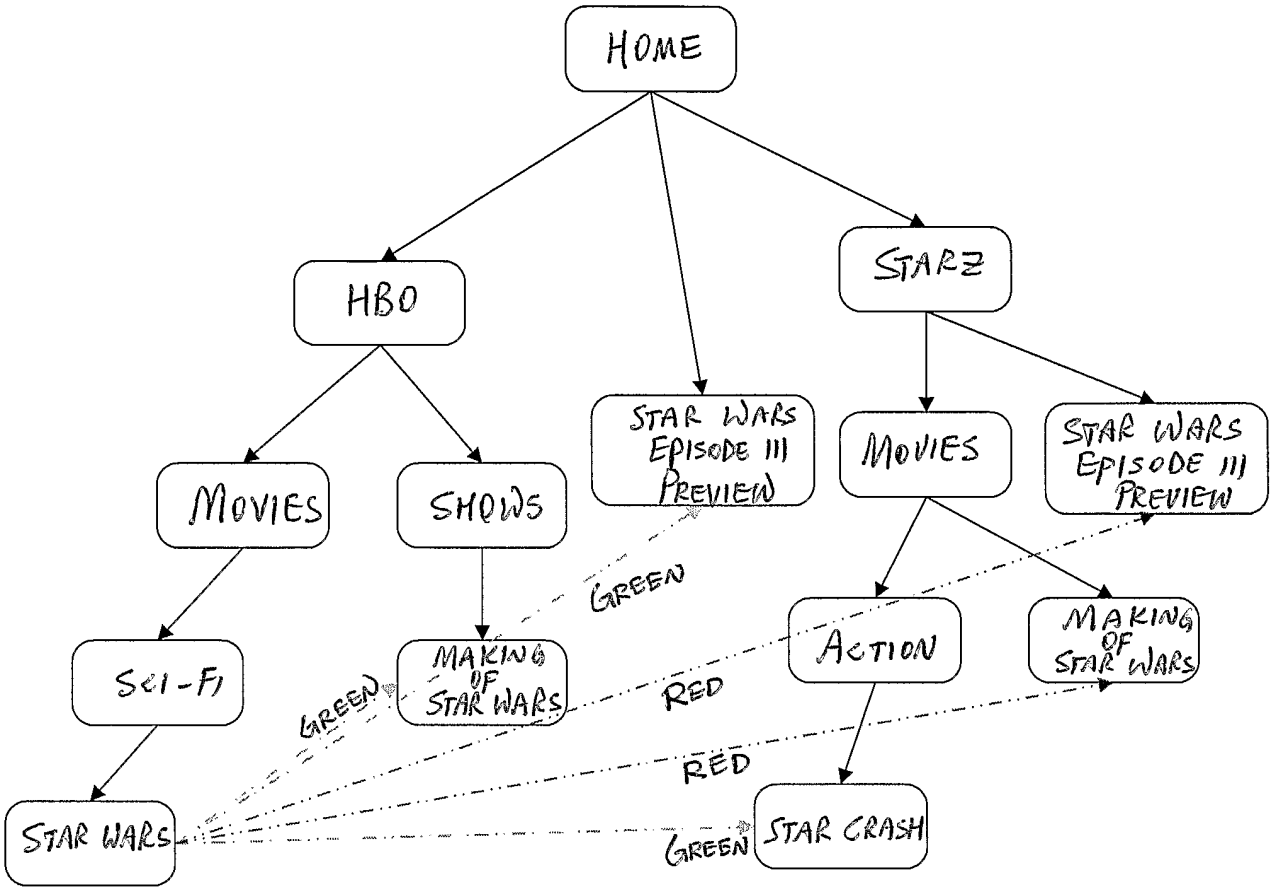
To make this work short-term, all possible query-result positions must be present in

shelf brick pages. If we need more than the number we made visible, we add invisible positions for the remaining positions.

Addendum: Definition of Shortest Path

The "shortest path" has the same meaning as in our first resolution attempt. There may be several nodes that can display an entity with the intended usage keywords. For example, a primary movie page could exist as Starz/movies/detail.svg and HBO/movies/description.svg.

The "closest" node containing this tag will be used as a hyperlink target. "Closest" refers to the node with the nearest common parent to the current node. For example:



In the above diagram, suppose the current node was "Star Wars" on the lower left. Related shows and their target nodes might be. Green arrows indicate the chosen targets; red arrows indicate possible targets that were eliminated.

- **Star Crash**
  - Only appears once
  - Choose that occurrence

- **Making of Star Wars**
  - Closest common parent is **HBO**
  - Choose **Making of Star Wars** node under **HBO**

- **Star Wars Episode III preview**

- Closest common parent has two occurrences under it

Choose the one nearest the parent

(If multiple with same distance from that parent, choose the first found

WHAT IS CLAIMED IS:

1. A method of returning target scenes from a user link request, said method comprising the steps of:

receiving a user link request on a user interface;

comparing said received user link request to a plurality of entries in a lookup table;

for each entry in the plurality of entries in the lookup table that matches the received user link request, identifying a target scene that corresponds to the matched entry in the lookup table;

determining a closest one of the target scenes if multiple entries in the lookup table match the user link request; and

transitioning to the closest target scene.

2. The method of claim 1, wherein a scene corresponds to a node, said scene being defined by a path to the node.

3. The method of claim 2, wherein the user link request occurs at a node.

4. The method of claim 3, wherein the closest target scene is determined by comparing a path of the node corresponding to the user link request with a path of each of the nodes corresponding to the target scenes.

5. The method of claim 4, further comprising:

selecting a target scene having the path with a longest common prefix with the path of the node corresponding to the user link request.

6. The method of claim 5, wherein if multiple target scenes have a same length common prefix, then

selecting the target scene having a fewest number of prefixes beyond the common prefix.

7. The method of claim 6, wherein if multiple target scenes have a same number of prefixes beyond the common prefix, then

selecting one of the multiple target scenes.

8. The method of claim 1, wherein a user interface corresponds to a home entertainment system.

9. The method of claim 1, wherein the user link request is received via a user input device.

10. The method of claim 9, wherein the user input device is a free space pointing device.

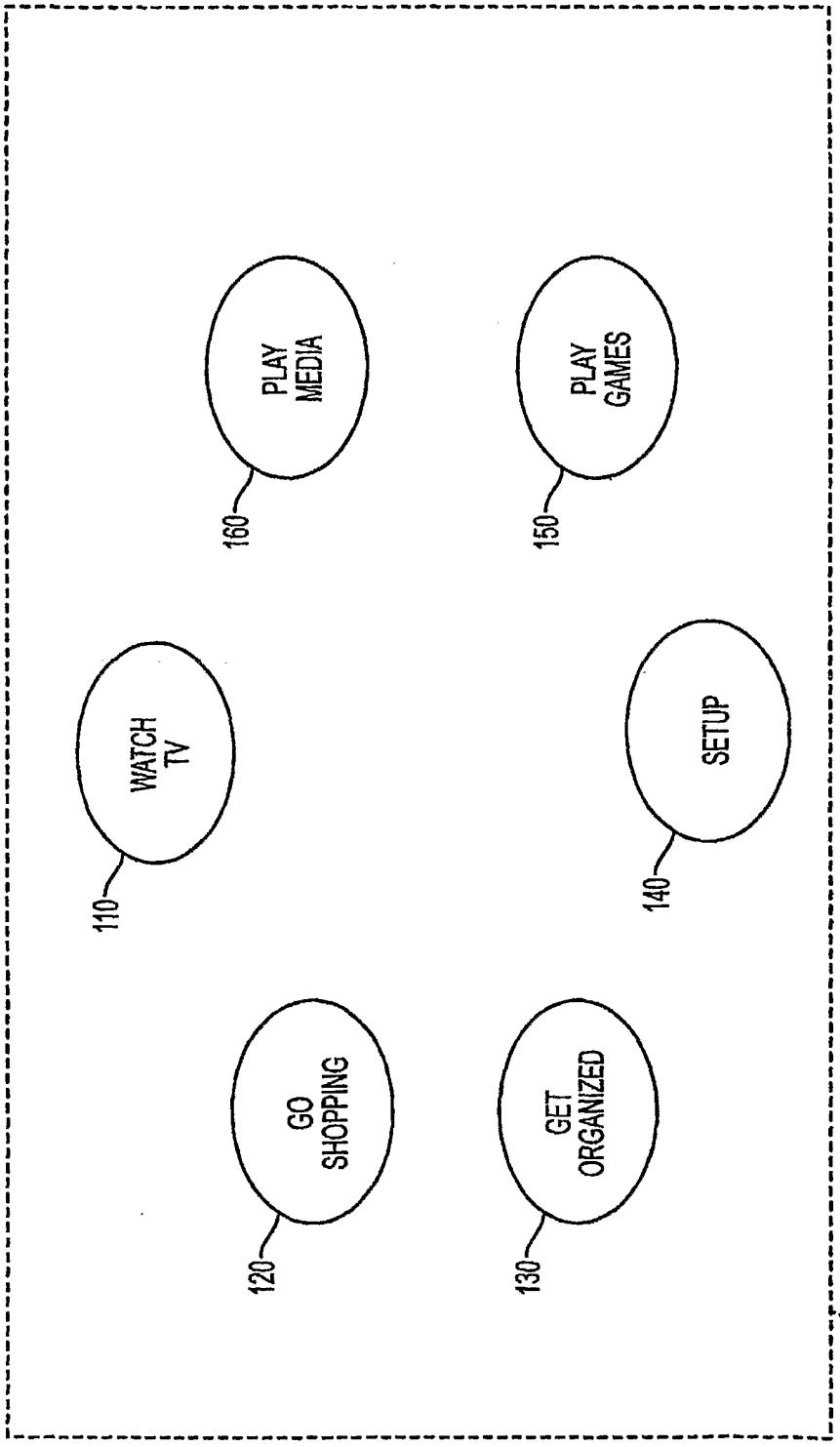

PLAY MEDIA
160
PLAY GAMES
150
WATCH TV
110
SETUP
140
GO SHOPPING
120
GET ORGANIZED
130
100

FIG. 1

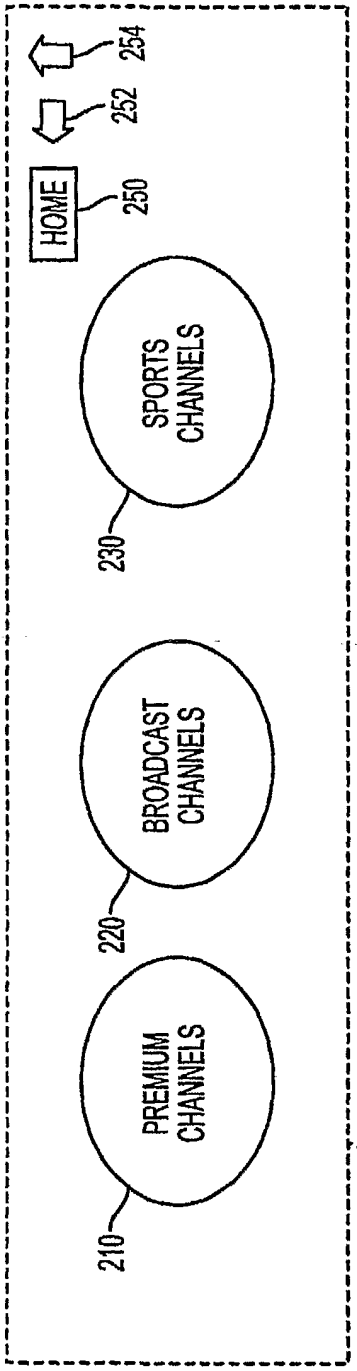

254
252
HOME
250
SPORTS CHANNELS
230
BROADCAST CHANNELS
220
PREMIUM CHANNELS
210
200

FIG. 2

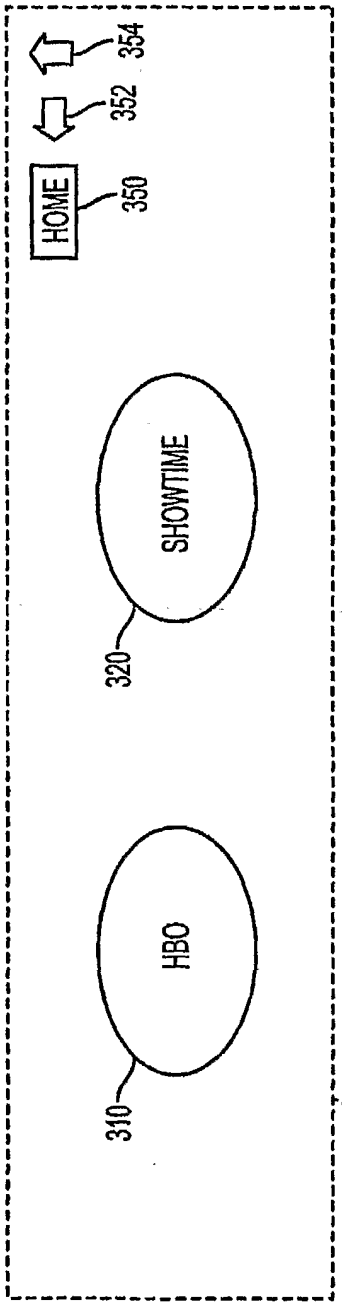

354
352
HOME
350
SHOWTIME
320
HBO
310
300

FIG. 3

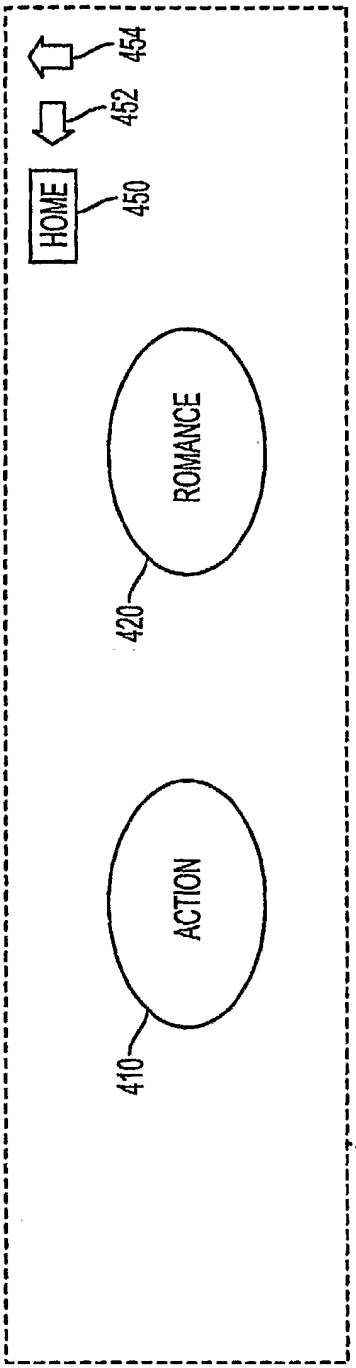

454
452
HOME
450
ROMANCE
420
ACTION
410
400

FIG. 4

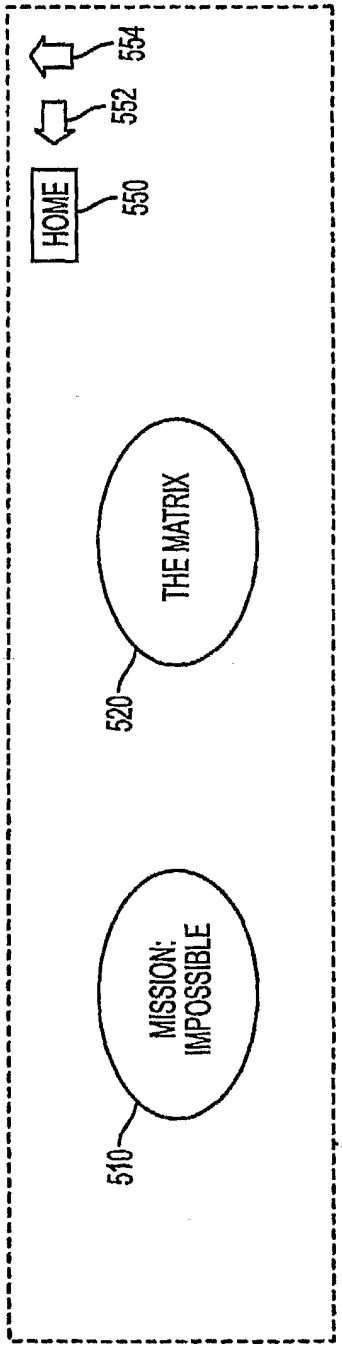

554
552
HOME
550
THE MATRIX
520
MISSION:
IMPOSSIBLE
510
500

FIG. 5A

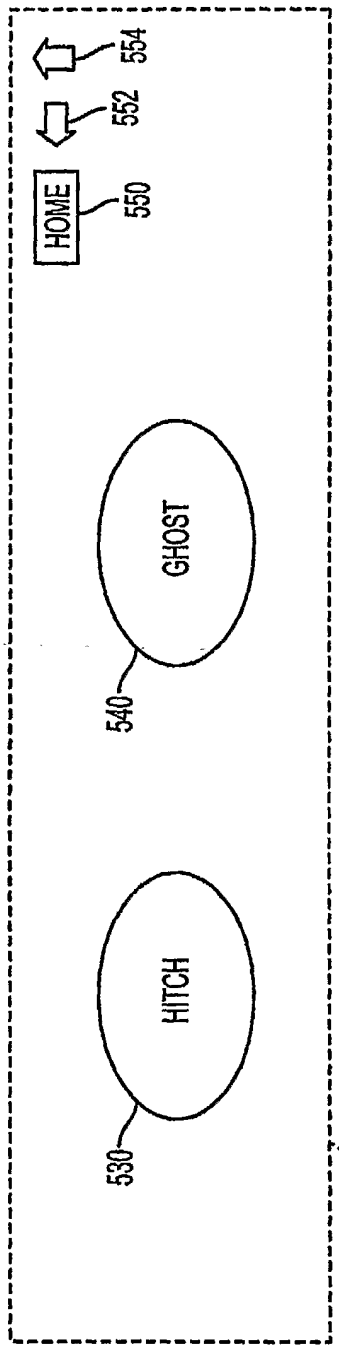


FIG. 5B

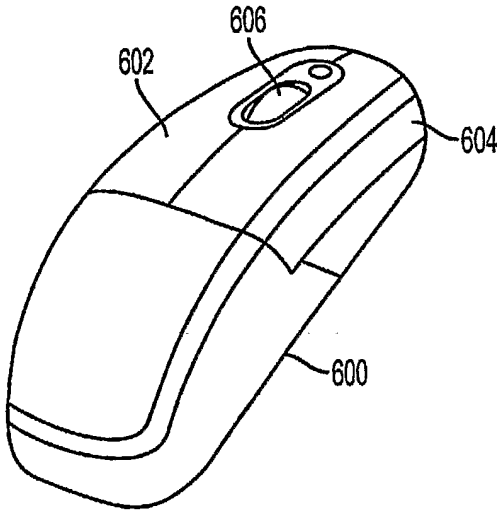


FIG. 6

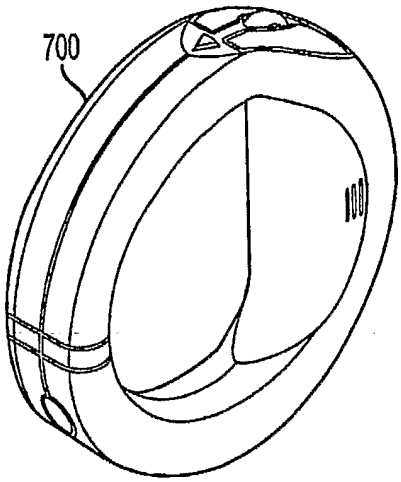


FIG. 7A

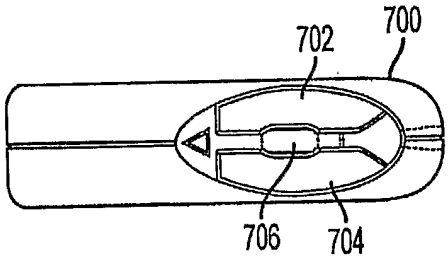


FIG. 7B

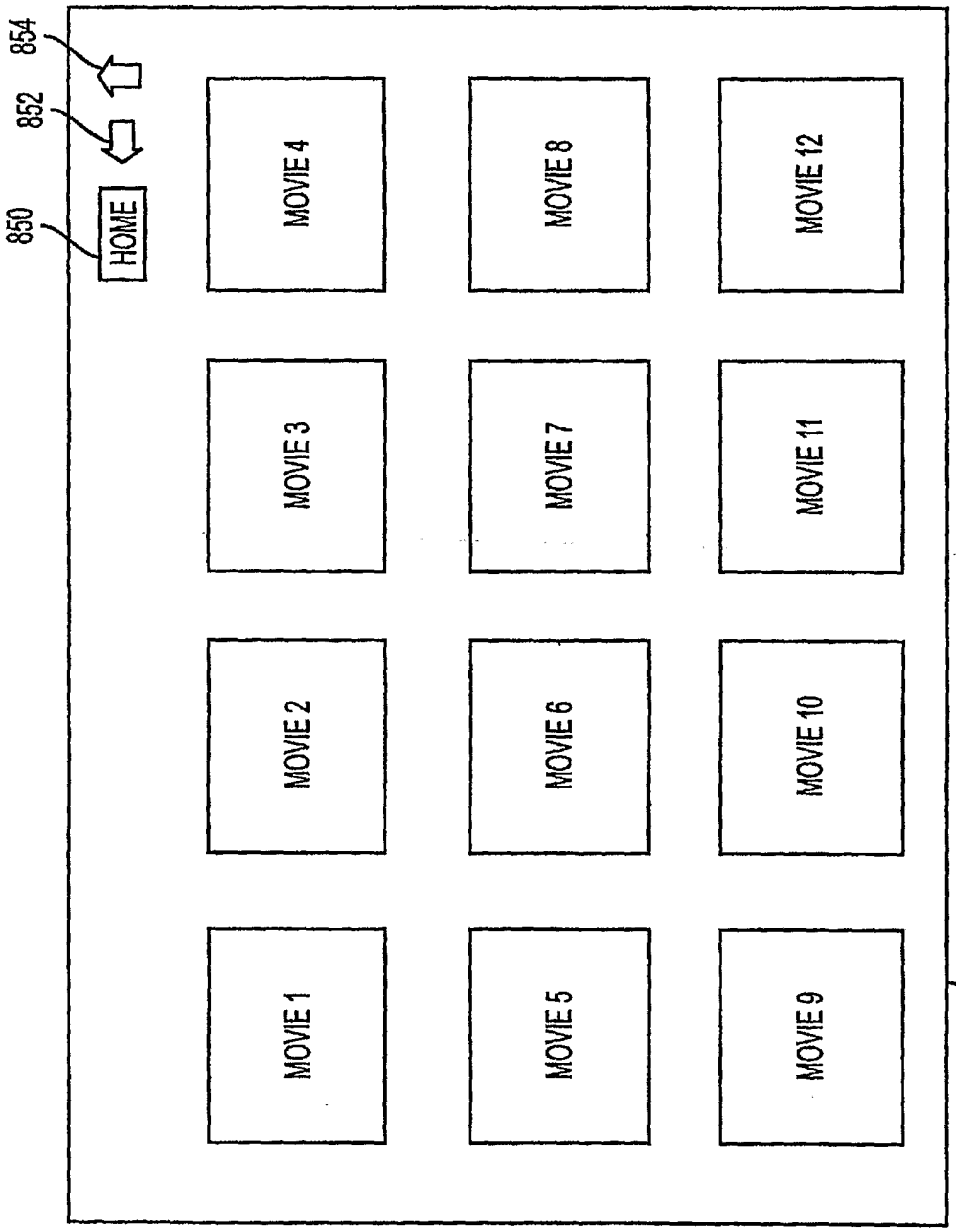

854
852
850
HOME
MOVIE 1
MOVIE 2
MOVIE 3
MOVIE 4
MOVIE 5
MOVIE 6
MOVIE 7
MOVIE 8
MOVIE 9
MOVIE 10
MOVIE 11
MOVIE 12
800

FIG. 8A

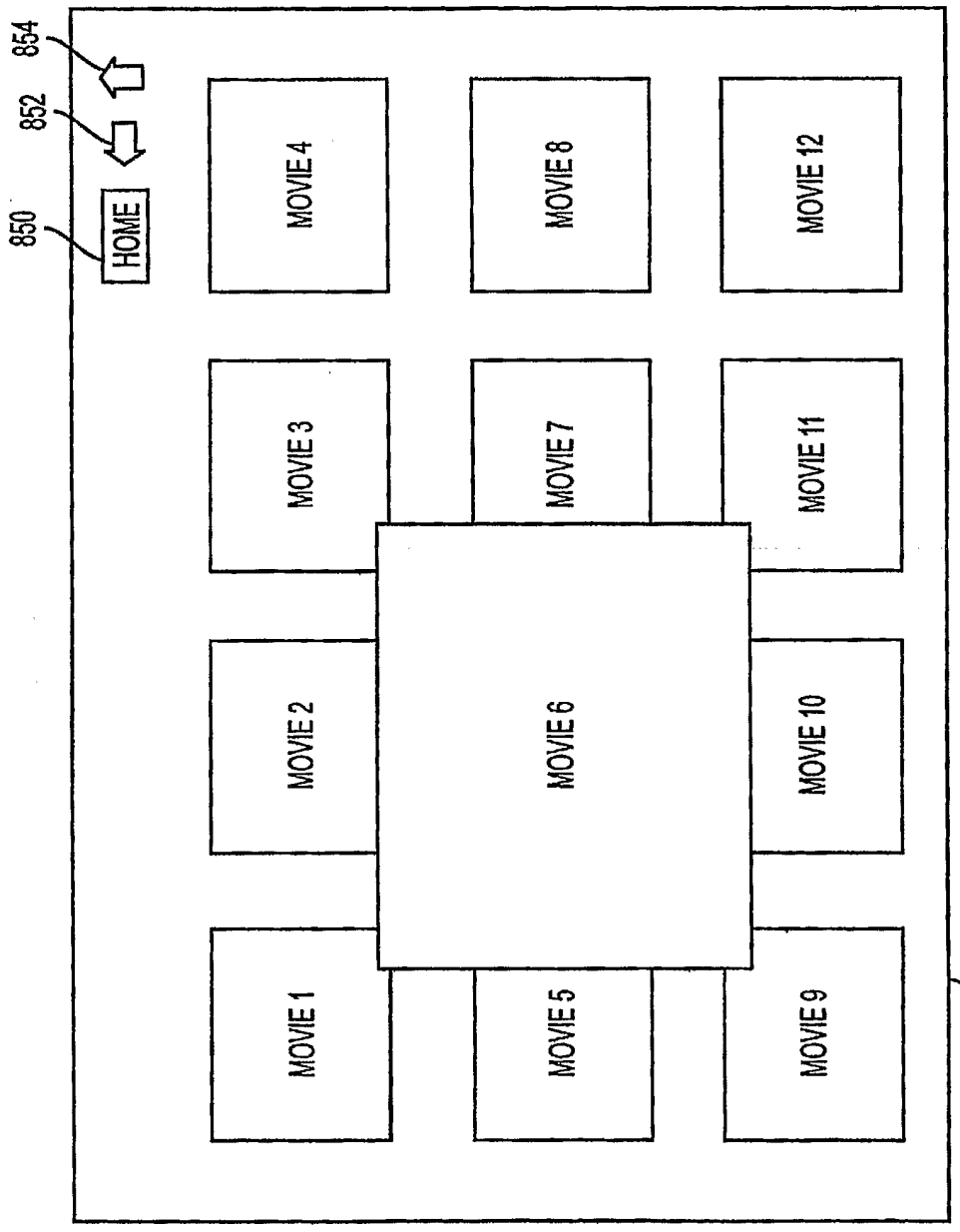

854
852
850
HOME
MOVIE 4
MOVIE 8
MOVIE 12
MOVIE 3
MOVIE 7
MOVIE 11
MOVIE 2
MOVIE 6
MOVIE 10
MOVIE 1
MOVIE 5
MOVIE 9
800

FIG. 8B

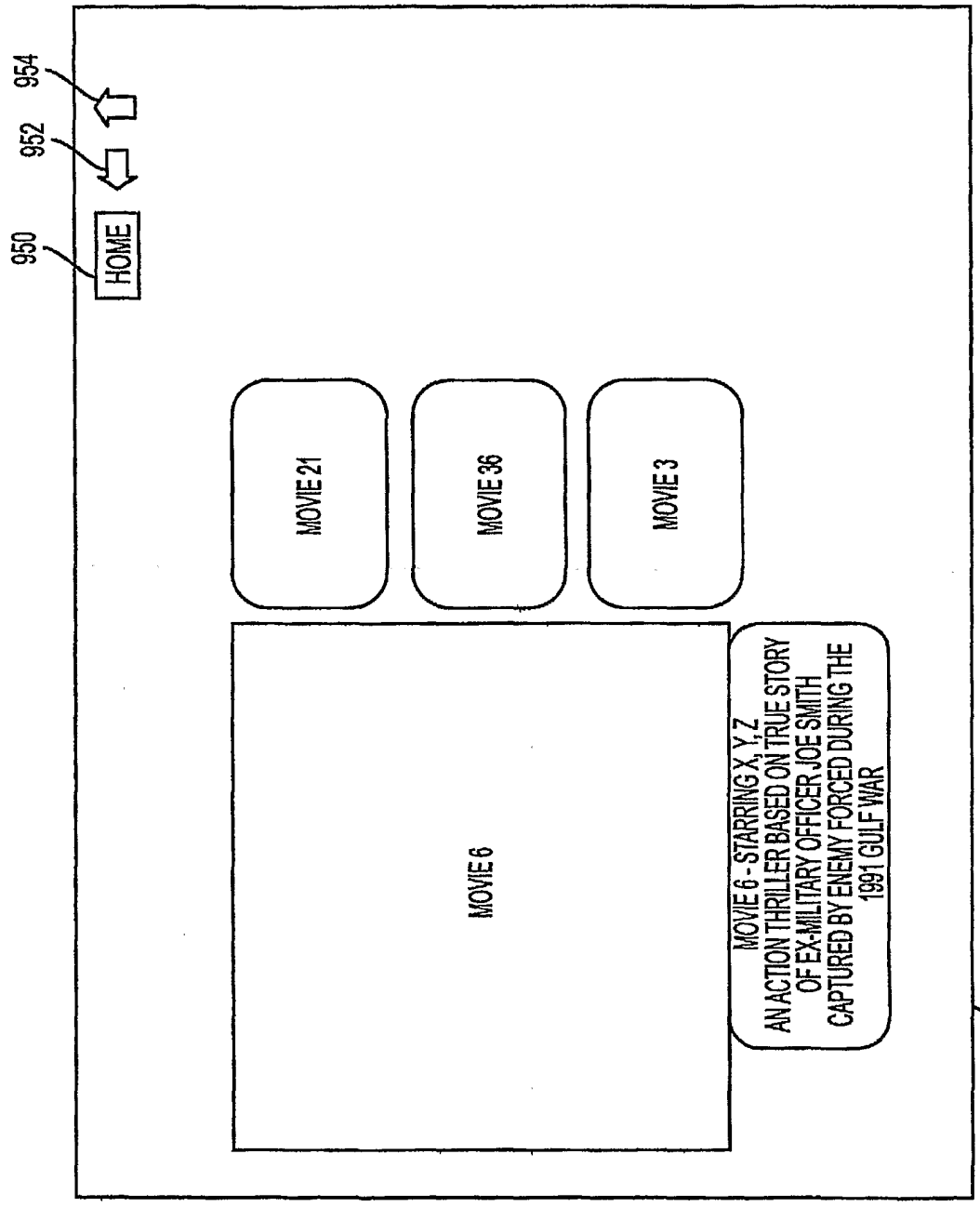

954
952
950
HOME
MOVIE 21
MOVIE 36
MOVIE 3
MOVIE 6
MOVIE 6 - STARRING X, Y, Z
AN ACTION THRILLER BASED ON TRUE STORY
OF EX-MILITARY OFFICER JOE SMITH
CAPTURED BY ENEMY FORCED DURING THE
1991 GULF WAR
900

FIG. 9

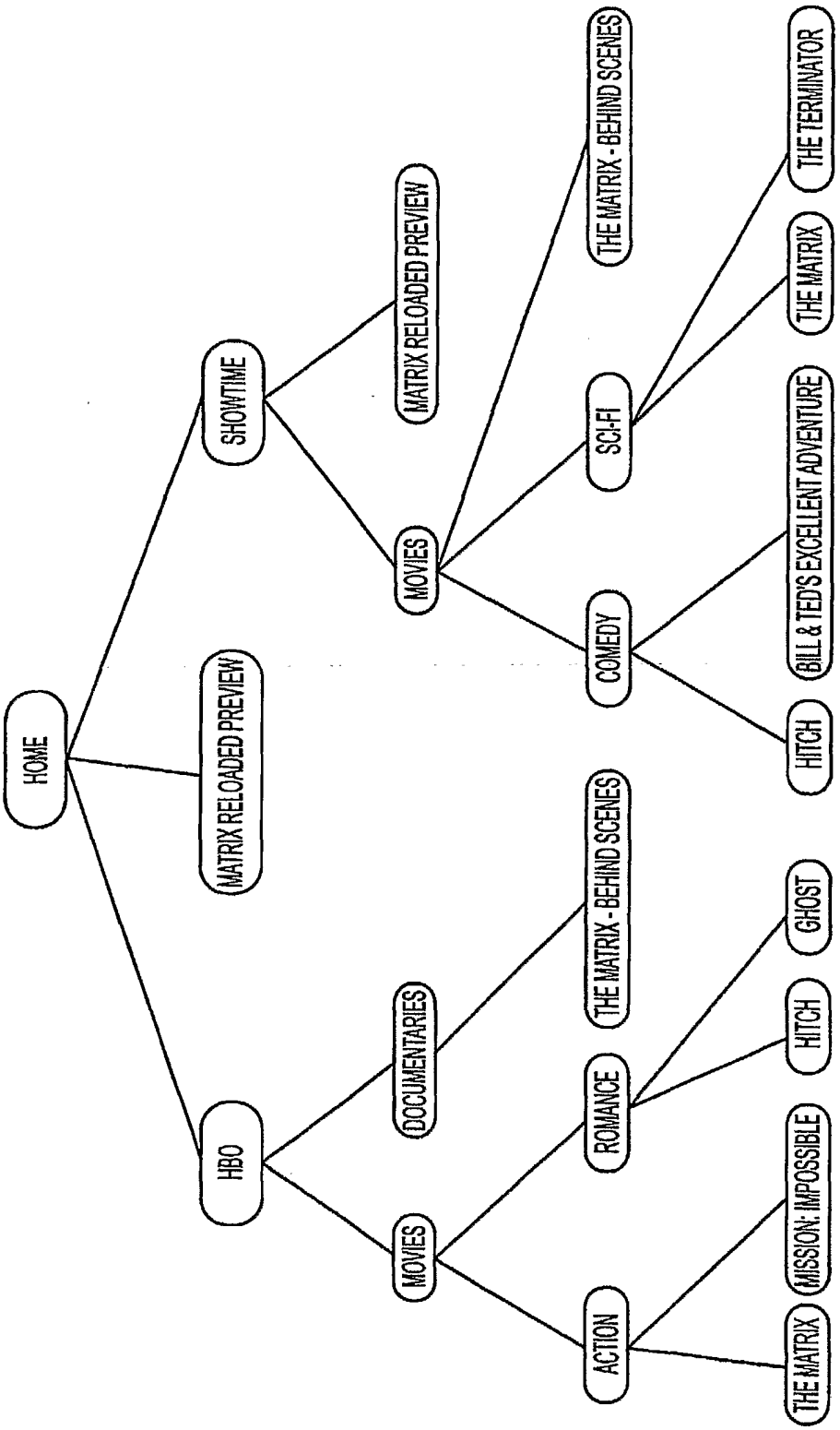

HOME
MATRIX RELOADED PREVIEW
HBO
MOVIES
ACTION
THE MATRIX
MISSION: IMPOSSIBLE
ROMANCE
HITCH
GHOST
DOCUMENTARIES
THE MATRIX - BEHIND SCENES
SHOWTIME
MOVIES
MATRIX RELOADED PREVIEW
COMEDY
HITCH
BILL & TED'S EXCELLENT ADVENTURE
SCI-FI
THE MATRIX
THE TERMINATOR
THE MATRIX - BEHIND SCENES

FIG. 10

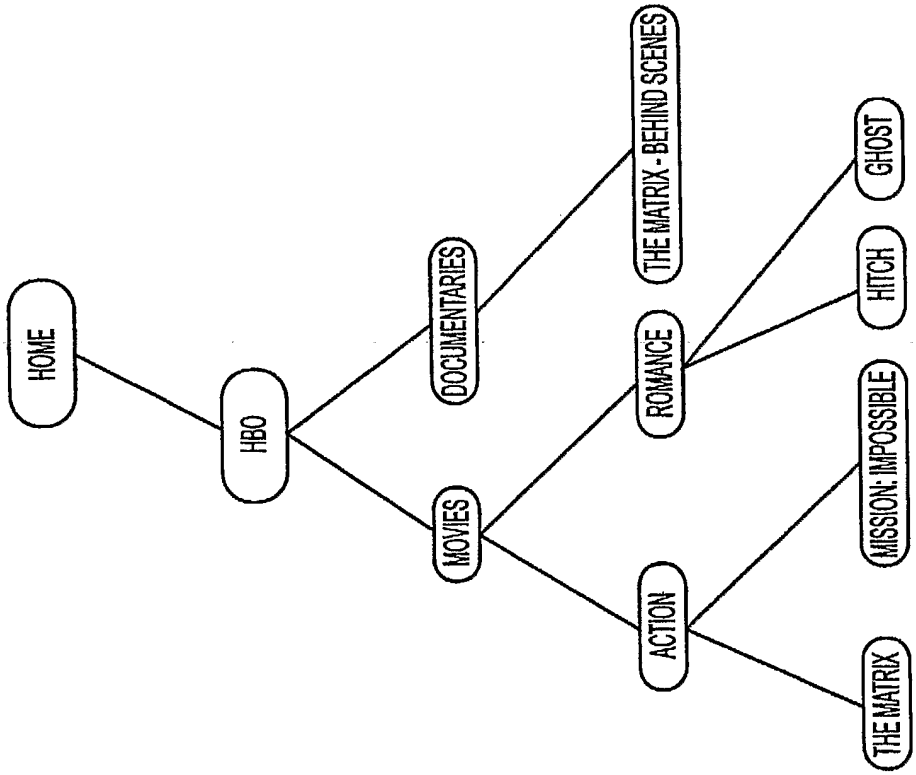

HOME
HBO
DOCUMENTARIES
THE MATRIX - BEHIND SCENES
MOVIES
ROMANCE
GHOST
HITCH
ACTION
MISSION: IMPOSSIBLE
THE MATRIX

FIG. 11A

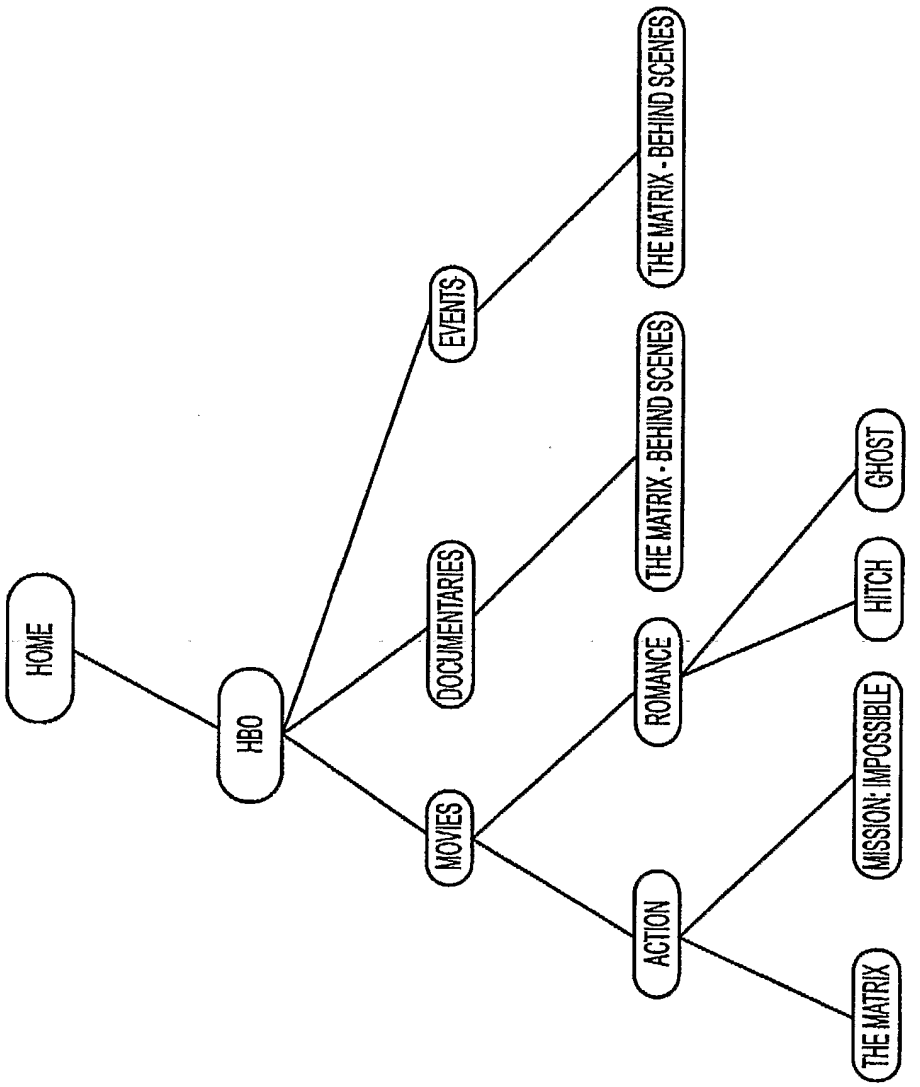

HOME
HBO
EVENTS
DOCUMENTARIES
MOVIES
THE MATRIX - BEHIND SCENES
THE MATRIX - BEHIND SCENES
ROMANCE
GHOST
HITCH
ACTION
MISSION: IMPOSSIBLE
THE MATRIX

FIG. 11B

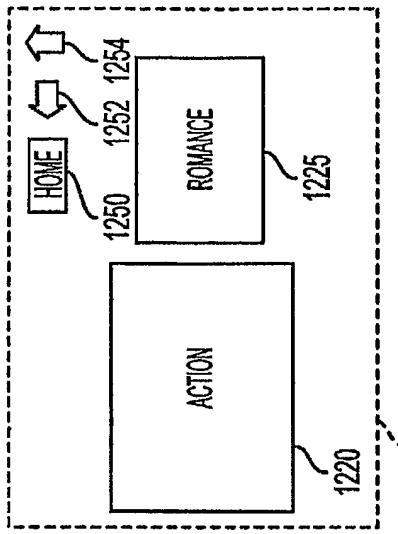


FIG. 12B

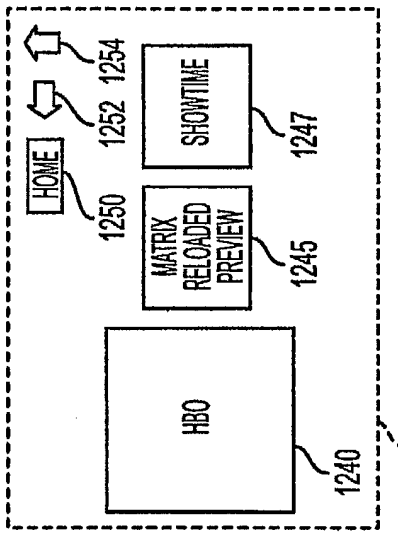


FIG. 12D

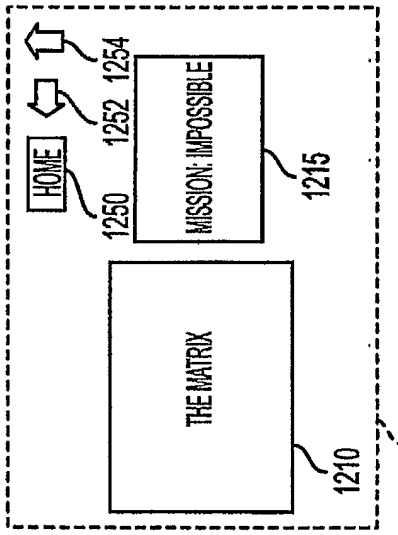


FIG. 12A

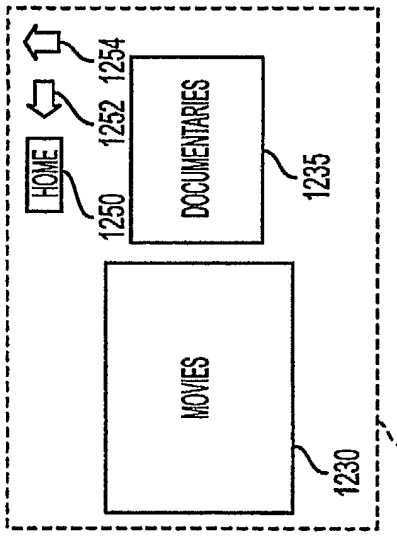


FIG. 12C

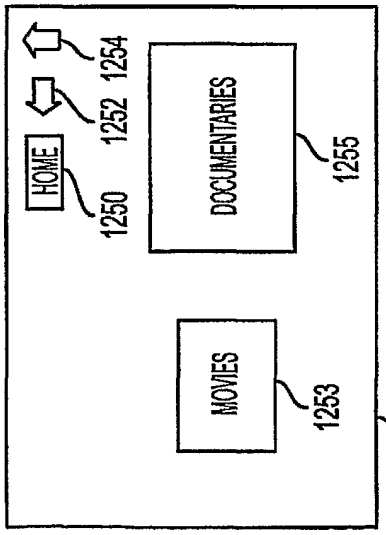


FIG. 12E

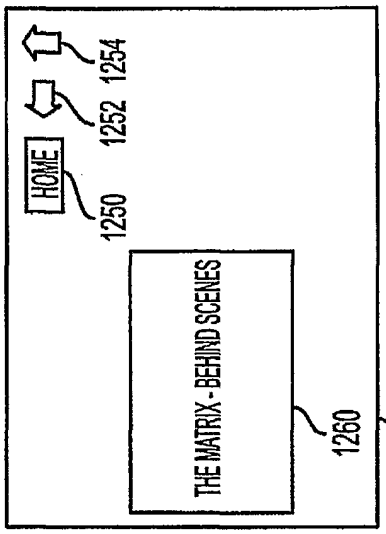


FIG. 12F

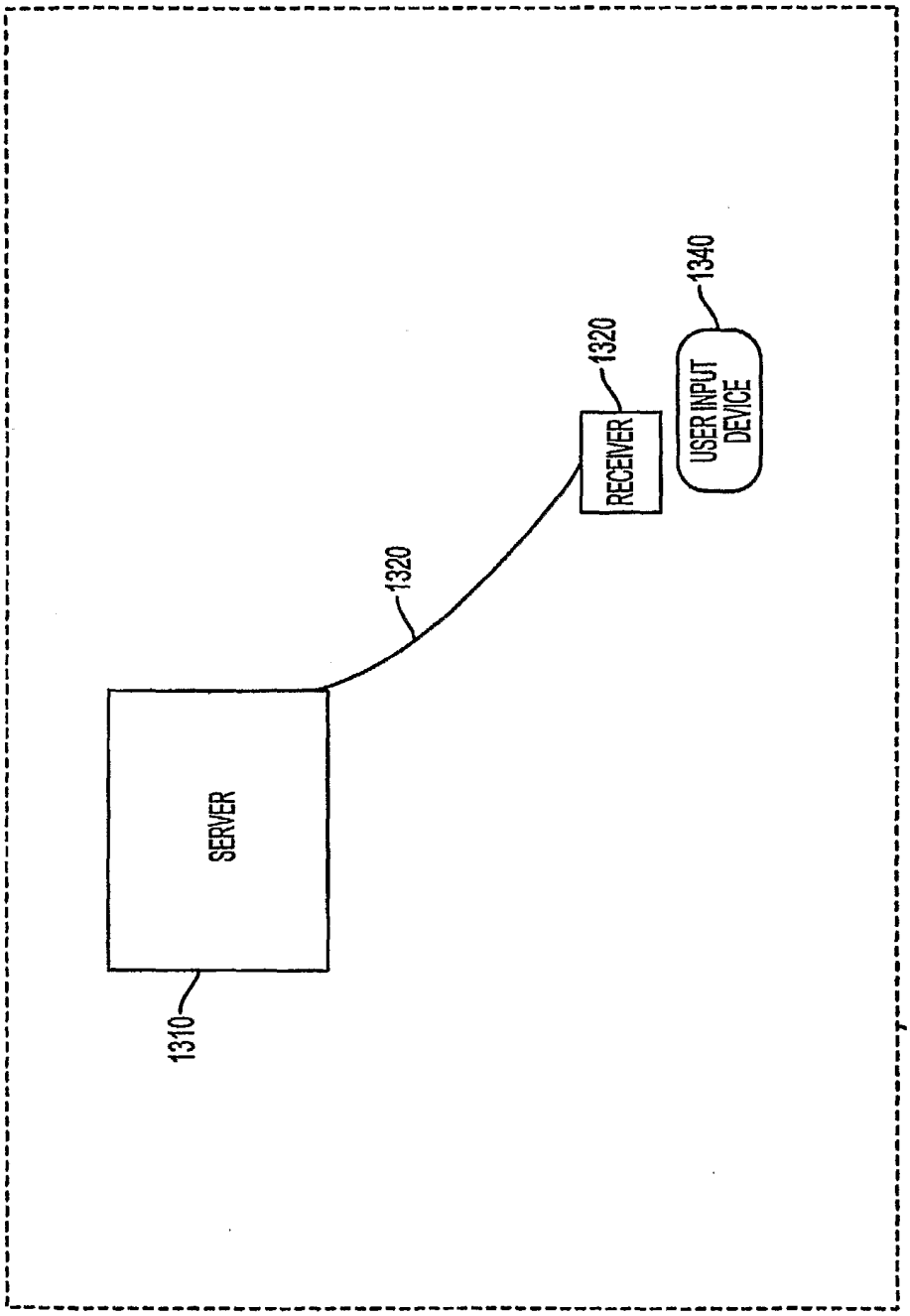


FIG. 13

16/16

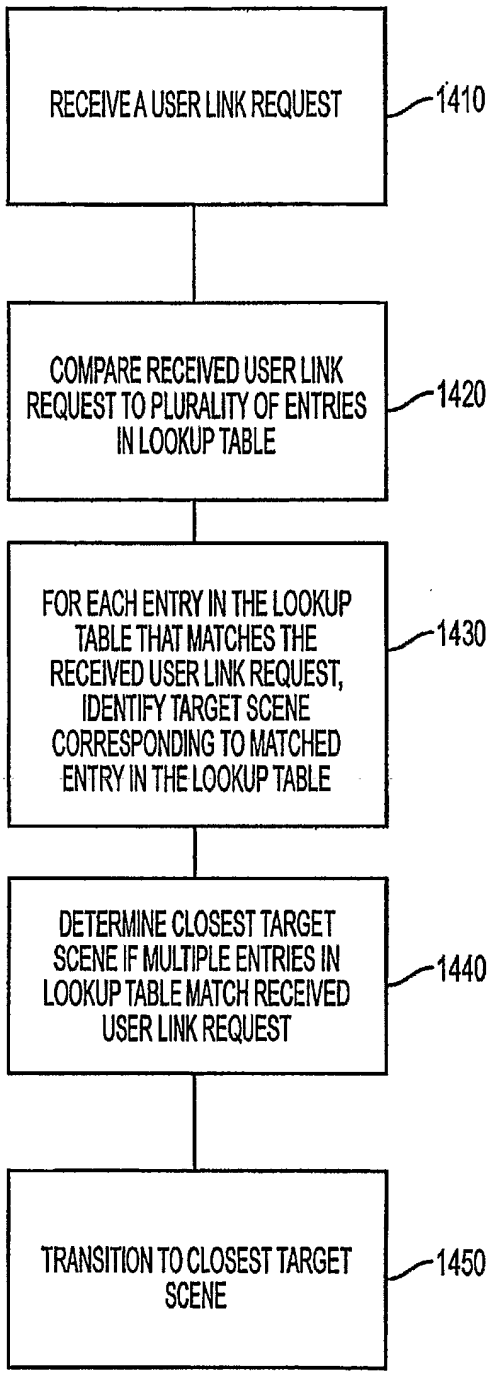

RECEIVE A USER LINK REQUEST
1410
COMPARE RECEIVED USER LINK REQUEST TO PLURALITY OF ENTRIES IN LOOKUP TABLE
1420
FOR EACH ENTRY IN THE LOOKUP TABLE THAT MATCHES THE RECEIVED USER LINK REQUEST, IDENTIFY TARGET SCENE CORRESPONDING TO MATCHED ENTRY IN THE LOOKUP TABLE
1430
DETERMINE CLOSEST TARGET SCENE IF MULTIPLE ENTRIES IN LOOKUP TABLE MATCH RECEIVED USER LINK REQUEST
1440
TRANSITION TO CLOSEST TARGET SCENE
1450

FIG.14