

- 1 -

WIRELESS COMMUNICATION DEVICE

This invention relates to the field of wireless
5 communication devices and specifically to man-machine
interfaces suitable for use with wireless communication
devices.

Man-machine interfaces (MMIs) are traditionally described
10 by a set of logical units which call functions in a
library on the device. The library provides a set of
functions which display user interface components on the
screen and by calling these library functions in certain
ways, and tying them together using program logic, the MMI
15 writer is able to render to the screen a graphical
depiction of the desired interface.

This approach has a number of disadvantages, for example,
using program logic to provide a rendered MMI requires
20 quite different skills to the skills required to describe
an ergonomic and aesthetically pleasing MMI.
Additionally, it is often awkward and undesirable to make
changes to the MMI once the communication device is
deployed in the marketplace and a new look-and-feel to the
25 MMI usually requires significant effort on the part of the
programmer to customise the library calls or the logical
units for the newly desired behaviour or appearance.

Therefore, it is desirable to try to discover an approach
30 to this problem that allows the writer of the logical
units to work in a fashion that is independent of the

- 2 -

designer of the MMI. This creates an "interface" between the two concerned parties, and allows for freedom to customise both sides of the "interface" at a late stage in production, or in fact once the wireless communication
5 device has been deployed.

According to a first aspect of the present invention there is provided a mobile communications terminal comprising a presentation entity and a plurality of logical entities;
10 the presentation entity comprising one or more presentation data sets and each logical entity having an associated software entity, the user interface for the mobile communications terminal being generated, in use, by querying one or more of the software entities to receive
15 data representing the state of the or each associated logical entity and then arranging the received logical entity data in accordance with a presentation data set.

The user interface for the terminal may be changed by
20 applying a further presentation data set to the received logical entity data. The series of software entities that are received may be altered and the further presentation data set applied to the altered logical entity data. The user interface for the terminal can be updated by
25 refreshing the data received from the one or more software entities.

The terminal may further comprise one or more display devices on which the terminal user interface can be
30 displayed. The terminal may further comprise user input means.

- 3 -

Preferably the terminal further comprises a co-ordinating entity that, in use, determines the software entities to be queried, receives the logical entity data from those software entities and applies a presentation data set to the received data to create a user interface data set. The terminal may further comprise a rendering entity, and, in use, the co-ordinating entity may send the display data set to the rendering entity, the rendering entity transforming the user interface data set such that it can be displayed. The terminal may further comprise a control entity which, in use, activates a terminal function in response to a specific event. In particular, the specific event may cause the control entity to execute a script. A specific event may be the user activating the user input means, or a variable, such as the time or date, reaching a specific value. The presentation data set may additionally comprise translation data.

According to a second aspect of the present invention there is provided a method of operating a mobile communications terminal, the method comprising the steps of: (a) generating one or more data items representing one or more logic entities within the terminal by querying the one or more logic entities; (b) applying a presentation data set to the generated data items to generate a user interface data set for the terminal.

Additionally the method may comprise the additional step of applying a translation data set to the generated data items before carrying out step (b). The method may also

- 4 -

comprise the additional step of (c) rendering the user interface data set and sending the results to a display device. Additionally, a presentation data set or a translation data set may be compiled into a binary format
5 and transmitted to the terminal.

The invention will now be described, by way of example only, with reference to the following Figures in which:

10 Figure 1 shows a schematic depiction of a wireless communication device according to the present invention;
Figure 2 shows a schematic depiction of the operation of the wireless communication device shown
15 in Figure 1;
Figure 3 shows a flowchart that outlines the operation of the engine;
Figure 4 shows a flowchart that describes the functioning of an actor following a request from the
20 engine;
Figure 5 shows a flowchart that describes the operation of the renderer;
Figure 6 shows a flowchart that describes the function of the agent;
25 Figure 7 shows a flowchart describing the process by which a MMI can be authored or modified ; and
Figure 8 shows a The Binary code compilation method is described in the form of a class diagram.

30 Figure 1 shows a schematic depiction of a wireless communication device 100 according to the present

- 5 -

invention. The device 100 comprises antenna 110, display screen 120, input interface 130, processor 140, storage means 145, operating system 150 and a plurality of further application programs 155.

5

Figure 2 shows a schematic depiction of the operation of the wireless communication device 100 shown in Figure 1. Engine 160 is in communication with message-based interface 165 that enables data to be sent and received from other system components. A resource manager 190 manages the storage of a shots entity 192, translation transform entity 194 and presentation transform 196 and it co-ordinates the passing of data from these entities to the engine 160. A collection of shots constitute a scene.

10 A shot may refer to static data or to dynamic data which will initiate an actor attribute query. The agent 200 passes updates to the resource manager and update notifications to the engine 160 via the interface 165. A renderer 170 receives a range of media elements, images, sounds etc from the resource manager 190. In an alternative implementation, multiple renderers may be used for different media types, such as audio content. The invention is also applicable to mobile devices with multiple screens, in which case multiple display renderers may be used. The renderer also receives renderer content from and sends user input data to the engine 160. The engine is also in communication with a plurality of actors 180; for the sake of clarity only actors 181, 182, 183, 184 are shown in Figure 2 but it will be appreciated that

25 a greater or lesser number of actors could be in communication with the interface 165.

30

- 6 -

The actors 180 represent the logical units of the wireless communication device such as, for example, the display screen, the renderer, input interface, power saving hardware, the telephone communications protocol stack, the plurality of further application programs, such as a calendar program. The renderer 170 is a computer program responsible for accepting an object description presented to it and converting that object description into graphics on a screen. The engine 160 has a number of functions that include: requesting and registering to receive updates to data from the actors 180; reading an object-based description of the data to query (which is referred to as a shot); taking data received from the actors 180 and placing the data into a renderer-independent object description of the desired MMI presentation (called a take); translating the renderer-independent object description into a new language, for example German, Hebrew, Korean, etc., as a result of the application of a translation stylesheet; and taking the translated renderer-independent object description and converting the data into a renderer-dependent object description as a result of the application of a presentation stylesheet. The agent is a further program 190 responsible for receiving communications from other entities and converting information received from those entities into requests for updates to actors, scripts, translation transforms, or presentation transforms. A script is the full collection of scenes and shots that make up the behavioural layer of an MMI. A shot comprises one or more spotlights, with a spotlight comprising zero or more actor

- 7 -

attribute queries. A spotlight without an actor attribute query constitutes a piece of content which is static before application of the presentation or language transform. An example of a basic user interface
5 comprising one scene and a number of shots is given in Annex A below.

The operation of the system described above with reference to Figure 2 will now be summarized. Figure 3 shows a
10 flowchart that outlines the operation of the engine 160. At step 300, the engine informs itself of the existence of installed actors by referring to a resource list installed alongside the script. At step 310, each actor establishes communication with the engine by registering with it. If
15 communication has not been established with all the actors then step 310 returns to step 300; if communication has been made with all the actors then at step 320 the engine loads a shot from the shot entity 192. The engine is set to first load a predefined scene (the start-up screen)
20 with its constituent shots.

During step 330 the engine 160 assesses and interprets the shot content data in order to determine which actors it will need data from. In step 340 the engine requests data
25 from one or more of the plurality of actors 180 that were identified in the shot content data. During step 350 the engine waits to receive the data from the actors. When all of the requested actors respond then the engine proceeds to step 360; otherwise if one or more of the
30 requested actors fail to respond, for example before a timer expires, then the engine returns to step 340 and

- 8 -

additional requests are sent to the actor(s) that have not responded.

The engine then processes the received data to form a take
5 during step 360 which is formatted by the application of a
translation stylesheet at step 370 and a presentation
stylesheet at step 380. The result of these various steps
is an object description that can be understood and
implemented by the renderer 170 and the final step 390 of
10 the process is to transmit the object description from the
engine to the renderer. The renderer will process the
object description, fetch associated referenced graphic or
multimedia content from the resource manager and display
or otherwise output the MMI defined within the object
15 description to the user.

Figure 4 shows a flowchart that describes the functioning
of an actor 180 following a request from the engine. At
step 440, the engine establishes communication with the
20 actor and the actor waits at step 410 in order to receive
a request for data from the engine. If the request from
the engine is valid then the actor proceeds from step 420
to step 430 and formulates a reply to the received
request. If the request is not valid then the actor
25 returns to step 410. The formulated reply will be sent to
the engine at step 440: if at step 450 the request is now
complete then the actor will return to step 410 to await a
further data request; otherwise the actor will wait for
the data to change (for example a decrease in battery
30 charge level) at step 460 before returning to step 430 to
generate a new reply to be sent to the engine.

- 9 -

Figure 5 shows a flowchart that describes the operation of the renderer 170. Once communication has been established with the engine at step 510 then the renderer waits for renderable object description data to be received from the engine (see above) at step 520. When suitable data is received then the data is rendered on the display screen 120 at step 530 and the renderer returns to step 520.

Figure 6 shows a flowchart that describes the function of the agent. The agent establishes communication with the engine in step 600 and then the agent waits to receive updates from the communications network at step 610. If it is desired to change one or more of the actors, translation stylesheet, presentation stylesheet or shots (these can be referred to as "Alterable Entities"), the agent is able to receive network communication from other entities (for example network or service providers, content providers, terminal manufacturers, etc.) containing alterations, additions or removals of an alterable entity. At step 620, the agent examines the received data to ensure that it is an alterable entity update. If so, at step 630 the alterable entity update is passed to the resource manager 190 in order that the appropriate entity is replaced with the updated entity and the entity update is also notified to the engine. If the data received is not an alterable entity update then the agent will discard the received data and will return to step 610 to await the reception of further data from the network.

- 10 -

The agent may initiate the downloading of an alterable entity update in response to a user action or at the prompting of the engine or resource manager (for example, an entity may have been in use for a predetermined time and it is required to check for an update or to pay for the right to continue to use it). Alternatively, updates may be pushed to the agent from a server connected to the terminal via a wireless communications network. To maintain the security and integrity of the terminal, it is preferred that the agent validates downloaded updates against transmission errors, viruses or other accidental or malicious corruption before passing the updates to the resource manager. Additionally, the agent may comprise DRM (digital rights management) functionality, which may include checking that received content has been digitally signed with an originating key that matches a receive key stored within the mobile device. A successful match results in proceeding with installation; an unsuccessful match may result in rejection, or installation of the update with limitations imposed, such as the update being un-installed after a limited period of time or installing the update with restricted functionality. The agent is also capable of prompting the removal of MMI content and/or alterable entities from the resource manager. Content may be removed, for example, after having been installed for a certain period of time, in response to a server command or a user input, or in order to make room for new content in the resource manager, etc..

Although most terminal (and thus actor) functionality will generally be incorporated at the time of manufacture, the

- 11 -

invention enables the addition of extra functionality, for example through the connection of a plug-in device such as, for example, a modem for an additional communications network or a non-volatile storage device. In this case, the actor software associated with the plug-in device, which may conveniently be uploaded from the device along a serial connection at the time of attachment, is installed into the actor collection, and a message is sent to the engine to register the new actor. Alternatively, the plug-in device may itself contain processing means able to execute the actor functionality, and communication between the engine and plug-in actor is achieved over a local communications channel. Appropriate de-registration will occur in the event of removal of the plug-in device.

User input events may come from key presses, touchscreen manipulation, other device manipulation such as closing a slide cover or from voice command input. In the latter case, a speech recognition actor will be used to translate vocal commands into message commands sent to the engine. It is well known that speech recognition accuracy is enhanced by restricting the recognition vocabulary to the smallest possible context. In this invention, each scene that has a potential voice input has an associated context. The context may be conveniently stored as part of the presentation transform entity, and transmitted to the speech recognition actor along with the renderer content for the display or other multimedia output.

The present invention greatly reduces the effort and complexity required to develop a new MMI (and also to

- 12 -

modify an existing MMI) when compared with known technologies. Figure 7 shows a flowchart describing the process by which a MMI can be authored or modified. In step 700 the new MMI is defined and created using an
5 authoring tool running on a personal computer or similar workstation. The output of the authoring tool is a description of the user interface in a mark-up language that is defined by a set of XML schema. As most current mobile communications terminals have significant
10 limitations to their storage capacity and processing power, in step 710 the mark-up language is compiled into a set of serialized binary-format objects. These objects can then be further processed during step 720 to provide a delivery package that can be placed on a server ready for
15 distribution to the mobile terminal.

At step 730 the MMI delivery package is transmitted to the mobile terminal, using for example, a data bearer of a wireless communications network where the package is
20 received by the radio subsystem in the mobile terminal (step 740). The MMI delivery package is then unwrapped by the agent at step 750 to recreate the binary files. These files are then validated and installed within the resource manager of the terminal for subsequent use (step 760).
25 Thus when the engine requires one of the MMI elements, such as a translation stylesheet for example, the newly downloaded style sheet can be passed to the engine (step 770) for processing before being sent to the renderer to be displayed to the user. (step 780). This technique also
30 enables subsequent updates to be supplied to a mobile terminal in a very simple fashion. The updated entities

- 13 -

can be compiled, packaged and transmitted, and the agent will ensure that only the newly received entity will be downloaded onto the terminal and that the entity to be replaced is deleted. It will be understood that any
5 convenient means of delivery of MMI packages may be used with this invention, including wireless and wired communications and plug-in storage media.

A terminal may store multiple MMI data sets. One MMI data
10 set may be used across all regions of the user interface, or multiple MMI data sets can exist concurrently in different regions of the user interface, allowing the user to navigate between different data sets. For example, each region will be dedicated to a different user function or
15 interest, such as shopping, news or control and configuration of the terminal. The MMI used for each region may be updated, inserted, activated, replaced or deleted independently of the others. It is also possible to update, replace, or delete one or more components of
20 the MMI data set, within any region or elsewhere within the user interface. When a new MMI data set is adopted, it may be selected to include either the behavioural functionality or the presentation layer, or both.

25 The terminal may comprise a control entity that can control the local operation of the terminal. This includes both the initiation of a simple function, such as making a phone call or activating a backlight, or a more complex function, such as causing a calendar data
30 synchronisation with a remote server. The control entity may be activated through the user input means or when

- 14 -

certain conditions are met, for example an alarm being triggered at a pre-determined time. Preferably, the control entity is able to execute a script, which may be initiated from one or more points within the user interface. Such a script can itself be downloaded, updated, inserted and deleted. A script allows complex sequences of functionality to be initiated by any user behaviour in the user interface. The combination of changeable MMI data sets and scriptable control functions within the control entity allows both the appearance of the user interface and the control behaviour to be changed together or independently. A control entity may be included within a MMI data set.

As described above, the data objects that are transmitted to terminals in order to add or update a MMI are compiled from a mark-up language into binary code. The mark-up language uses a number of behaviour and presentation schemas to describe a MMI for mobile devices. The behaviour schemas referred to as the script comprise:

1. Reusable sets of strands which are threads of behaviour initiated by specific events in the phone;
2. A description of how each page is built up from a set of page fragments (scenes);
3. A description of how each page fragment is built up from a set of queries that can be addressed to the components represented by actors, in order to populate a page with dynamic content (shot);
4. A set of page transition conditions, that is the renderer/logic events that cause the MMI to move from

- 15 -

one page to another (scene change condition);

5. Page interrupt conditions, that is the renderer/logic events that cause a page context to be saved, interrupted and subsequently restored after a page sequence has completed (strand conditions); and
6. State transition machines for managing interaction between MMI events and logic events, for example describing how to handle an MP3 player when an incoming call occurs, and for allowing page content to be state-dependent (for example the background image of the page currently on display changing as a result of a new SMS message being received).

The presentation schemas comprise:

1. Transforms that describe how a presentation-free page fragment built by the MMI execution engine (within the portable device) can be converted into a presentation-rich format suitable for a specialised renderer (sets).
2. Transforms that describe how a language-neutral page fragment can be converted into a language-specific page fragment.
3. Transforms that describe how a presentation-free page assembled by the engine can be converted into a presentation-rich format for sending to a specialised renderer.

Additionally to the schemas described above, the mark-up language has the capability to handle and execute multimedia resources and files, including graphics,

- 16 -

animations, audio, video, moving banners etc.

The compilation of the mark-up language into a set of serialized binary-format objects provides a further
5 advantage in that the mark-up language does not need to be
parsed by the wireless terminal. This has very
significant implications for the design of the terminal as
the terminal will be able to execute commands in response
to user inputs more quickly (as each display update would
10 require several ML objects to be parsed into binary).
There will also be a saving made in the storage and memory
requirements for the terminal, as the mark-up language
text is less compact than the binary objects and there is
no longer a need to supply an XML parser to convert the
15 mark-up language into binary code. An implementation of
the binary format is shown in Figure 8. An example
hexadecimal listing resulting from the binary compilation
is shown below in Annex B.

20 A still further advantage of the present invention is that
the logic units that are represented by the actors are
separate from the MMI. Thus the designer of the logic
units does not need to know anything about the manner in
which the data provided by the logic units will be used
25 within the MMI (and similarly the MMI designer does not
need to know anything about the logic units other than
what data can be queried from them). This separation
provides a number of advantages, for example: enabling the
MMI to be changed rapidly if required (with the new code
30 being uploaded to the communication device via a network
entity if necessary); rewriting the MMI becomes a much

- 17 -

simpler task and it is possible to provide several different presentation stylesheets within a wireless terminal, thereby allowing users to have a choice of several different MMIs, each with display characteristics
5 of their own choosing.

It will be clearly understood that the present invention may be implemented within a wide range of mobile communication terminals, such as cellular radio telephones
10 (using 2G, 2.5G or 3G bearer networks), personal digital organisers having wireless communications capabilities (i.e. telephony modems, wireless or optical LAN connectivity, etc.), etc. and that the nature of the terminal or the manner of its communication should not
15 effect the use of the invention.

ANNEX A

```
5  <?xml version="1.0" encoding="UTF-8"?>

    <!DOCTYPE SCRIPTCHUNK SYSTEM
    "..\..\trigenix3engine\documentation\design and
    architecture\schema\script.dtd">

    <!-- Yann Muller, 3G Lab -->
10  <!-- T68 Calculator menu -->

    <SCRIPTCHUNK>

    <ROUTINE ROUTINEID="Calculator.Home"
    STARTINGSCENEID="Calculator.Home"
    TEMPLATECHANGECONDITIONSID="NestedRoutine">
15  <!-- Calculator Home -->

    <SCENE SCENEID="Calculator.Home" LAYOUTHINT="standard"
    STRANBLOCKID="standard">

        <SHOTIDS>

            <SHOTID>Calculator.Memory</SHOTID>
20      <SHOTID>Calculator.Operand1</SHOTID>

            <SHOTID>Calculator.Operand2</SHOTID>

            <SHOTID>Calculator.Result</SHOTID>

        </SHOTIDS>

        <CHANGECONDITIONS>
25      <CHANGECONDITION SCENEID="Organizer.Home">

            <INACTOREVENT ACTORID="keypad" EVENTID="no"/>

        </CHANGECONDITION>

        </CHANGECONDITIONS>

    </SCENE>
```

- 19 -

```
</ROUTINE>

<!-- Shots -->

<!-- Display of the calculator's memory -->
<SHOT SHOTID="Calculator.Memory">
5      <SPOTLIGHTDESCRIPTION KEY="Memory">
          <EVENTMAPS>
              <ACTORQUERY ACTORID="Calculator"
ATTRIBUTEID="Memory"/>
          </EVENTMAPS>
10     </SPOTLIGHTDESCRIPTION>
        </SHOT>
        <!-- Display of the first operand -->
        <SHOT SHOTID="Calculator.Operand1">
            <SPOTLIGHTDESCRIPTION KEY="Operand1">
15          <EVENTMAPS>
              <ACTORQUERY ACTORID="Calculator"
ATTRIBUTEID="Operand1"/>
          </EVENTMAPS>
            </SPOTLIGHTDESCRIPTION>
20     </SHOT>
        <!-- Display of the operator and second operand -->
        <SHOT SHOTID="Calculator.Operand2">
            <SPOTLIGHTDESCRIPTION KEY="Memory">
                <EVENTMAPS>
25          <ACTORQUERY ACTORID="Calculator"
ATTRIBUTEID="Operand2"/>
          </EVENTMAPS>
            </SPOTLIGHTDESCRIPTION>
```

- 20 -

```
</SHOT>

<!-- Display of the result -->

<SHOT SHOTID="Calculator.Result">
    <SPOTLIGHTDESCRIPTION KEY="Result">
5        <EVENTMAPS>
            <ACTORQUERY ACTORID="Calculator"
ATTRIBUTEID="Result"/>
        </EVENTMAPS>
    </SPOTLIGHTDESCRIPTION>
10 </SHOT>

<!-- Capabilities -->

<CAPABILITIES>
    <!-- attributes -->
    <CAPABILITY ID="Memory" TYPE="attribute">
15        <!-- the value of the memory -->
            <PARAMETER TYPE="decimal" NAME="Memory"/>
        </CAPABILITY>
    <CAPABILITY ID="Operand1" TYPE="attribute">
        <!-- The first number of the current operation -->
20        <PARAMETER TYPE="decimal" NAME="Number1"/>
    </CAPABILITY>
    <CAPABILITY ID="Operand2" TYPE="attribute">
        <!-- The second number and the operator -->
        <PARAMETER TYPE="string" NAME="Operator"/>
25        <PARAMETER TYPE="decimal" NAME="Number2"/>
    </CAPABILITY>
    <CAPABILITY ID="Result" TYPE="attribute">
```

- 21 -

```
        <!-- The result -->
        <PARAMETER TYPE="decimal" NAME="Result"/>
    </CAPABILITY>
    <!-- eventsin -->
5    <!-- eventsout -->
    </CAPABILITIES>
</SCRIPTCHUNK>
```

ANNEX B

5	00000000	0000	0600	0000	0100	0000	0200	0000	0300
	00000010	0000	0400	0000	0500	0000	0600	0000	0200
	00000020	0001	0000	0101	ffff	ffff	0000	0000	0000
	00000030	0400	0000	0100	0000	0200	0000	0300	0000
	00000040	0400	0000	0000	0000	0100	0000	0100	0000
10	00000050	0100	0000	0100	0000	0600	ffff	ffff	0000
	00000060	0000	0000	0000	0000	0200	0000	0100	0000
	00000070	0200	0000	0100	0000	0600	ffff	ffff	0000
	00000080	0000	0000	0000	0000	0300	0000	0100	0000
	00000090	0100	0000	0100	0000	0600	ffff	ffff	0000
15	00000a0	0000	0000	0000	0000	0400	0000	0100	0000
	00000b0	0300	0000	0100	0000	0600	ffff	ffff	0000
	00000c0	0000	0000	0000					
	00000c6								

CLAIMS

1. A mobile communications terminal comprising a
5 presentation entity and a plurality of logical entities;
the presentation entity comprising one or more
presentation data sets and each logical entity having an
associated software entity, the user interface for the
mobile communications terminal being generated, in use, by
10 querying one or more of the software entities to receive
data representing the state of the or each associated
logical entity and then arranging the received logical
entity data in accordance with a presentation data set.
- 15 2. A mobile communications terminal according to claim
1, wherein the user interface for the terminal can be
changed by applying a further presentation data set to the
received logical entity data.
- 20 3. A mobile communications terminal according to claim 1
or claim 2, in which the user interface for the terminal
can be updated by refreshing the data received from the
one or more software entities.
- 25 4. A mobile communications terminal according to claim 2
wherein the series of software entities that are queried
is altered and the further presentation data set is
applied to the altered logical entity data.
- 30 5. A mobile communications terminal according to any
preceding claim, in which the terminal further comprises a

- 24 -

display device on which the terminal user interface can be displayed.

6. A mobile communications terminal according to any
5 preceding claim, in which the terminal further comprises user input means.

7. A mobile communications terminal according to any
preceding claim, in which the terminal further comprises a
10 co-ordinating entity that, in use, determines the software entities to be queried, receives the logical entity data from the queried software entities and applies a presentation data set to the received data to create a user interface data set.

15
8. A mobile communications terminal according to claim 7, in which the terminal further comprises a rendering entity, and, in use, the co-ordinating entity sends the display data set to the rendering entity, the rendering
20 entity transforming the user interface data set such that it can be displayed.

9. A mobile communications terminal according to any preceding claim, in which the terminal further comprises a
25 control entity, the control entity, in use, activating a terminal function in response to a specific event.

10. A mobile communications terminal according to any claim 9, wherein the specific event causes the control
30 entity to execute a script.

- 25 -

11. A mobile communications terminal according to any preceding claim, in which the presentation data set additionally comprises translation data.

5 12. A method of operating a mobile communications terminal, the method comprising the steps of:

(a) generating one or more data items representing one or more logic entities within the terminal by querying the one or more logic entities;

10 (b) applying a presentation data set to the generated data items to generate a user interface data set for the terminal.

13. A method according to claim 12, the method comprising
15 the additional step of applying a translation data set to the generated data items before carrying out step (b).

14. A method according to claim 12 or claim 13, the method comprising the additional step of;

20 (c) rendering the user interface data set and sending the results to a display device.

15. A method according to any one of claims 12 to 14, wherein a presentation data set or a translation data set
25 is compiled into a binary format and transmitted to the terminal.