basic windows for the UDI begins at a start step **1302**. Then a set of application resources are passed to a Template .dll, as a step **1304**. The Template .dll comprises the resources within the UDI that gives the Zenu™ UDI its shape and controls position of the buttons. At a next step **1306**, a Theme .dll is loaded, as specified in an .ini file, or the like. The load Theme .dll stores all of the individual overrides of the Template in terms of its default appearance. At a next step, **1308**, the relevant "UDI window look" is applied, together with other window settings. The details of step **1308** are described below in connection with **FIG. 14**. Next, at a step **1310**, the buttons of the Zenu™ UDI are created as basic window definitions. The details of button creation are described below in connection with **FIG. 15**. Various settings of the buttons are then applied at a step **1312**. The details of the button settings are described below in connection with **FIG. 16**. At a next step, **1314**, the Zenu™ UDI waits for an event to occur. At this stage two events can occur; a button can be clicked, or a drop file unbutton event can occur. A "button click" is described below in connection with **FIGS. 17 and 18**. A "drop file unbutton event" is described below in connection with **FIG. 19**.

[0142] **FIG. 14** describes the details of a plain UDI window look and other window settings as introduced at step **1308**. This process starts at a step **1402**. Based on the availability of the template and theme information, a decision is made (step **1404**) to determine if the "main window look" is overridden in the theme file. If YES (i.e., the theme is going to be used), a region is created according to a "main UDI window look" in the theme, as shown at a step **1406**. Otherwise, information from the default template will be used to create the region, as shown at step **1408**. Next, at a step **1410**, the window pop-up position information is retrieved from the .ini file. In other words, the position where the Zenu™ UDI was last displayed is obtained, or a cursor relative position is determined. At a step **1412** the window is then registered as an application object with a operating system for tracking and access purposes. Thus, the operation is performed and the flow FIG. of **14** represents a "shell" of the window for the Zenu™ UDI. Next, at a step **1414**, the process flows to step **1310**, which is further described in connection with **FIG. 15**.

[0143] **FIG. 15** illustrates the flow in connection with "creating buttons" as introduced at step **1310**. The flow begins at a start step **1502**. Steps **1504** through **1518** access the template to determine the designated number of buttons that make up the Zenu™ UDI. For example, the Zenu™ UDI of **FIG. 8A** comprises **12** buttons; two rows of **6** buttons each. In essence, this figure represents the creation of a small window corresponding to each button and links them together to create a Zenu™ shell. Every time a new Zenu™ UDI session is initialized, i.e., the Zenu™ UDI is launched, small windows called "child windows" corresponding to each button of the Zenu™ UDI must be created to form the UDI, as shown at a step **1504**. A first button is processed as shown at a step **1506**. At a step **1508**, it is then determined whether the child window is a button. If YES, a UDI button is created and it is assigned a subclass as a child window, at a step **1510**. Next, a pointer to the button is stored in a list for future access, as shown at a step **1512**. If an additional child window is to be processed, as determined at a step **1514**, the next button is retrieved, as shown at a step **1516**. The process then flows back to step **1508**, otherwise the flow proceeds to step **1312** as shown by step **1518**. Step **1312** applies various settings to each button just created, the details of which are described in connection with **FIG. 16**.

[0144] **FIG. 16** further illustrated the process of step **1312** for applying various settings to the buttons created in **FIG. 15**. Thus, the steps illustrated in **FIG. 16** are performed for each button for which a region was defined in **FIG. 15**. The flow starts at a step **1602** and proceeds to get a first button for processing at step **1604**. At a step **1606** a theme name and parent template name are set for the button. Next, at a step **1608**, the button configuration string from the .ini file is set. Next, at a step **1610**, an application resource is set to the UDI template file. At a step **1612**, the configuration string from the button for the template file is loaded. At a step **1614**, the application resource is set to the UDI windows current theme file. At step **1616**, the configuration string from the button is loaded from the theme file. At a step **1618**, the theme settings string is merged with the template settings thereby overriding the values in the template setting string (i.e., theme settings are replaced by template settings).

[0145] At a step **1620**, the button configuration that was loaded from the .ini file is merged with the existing string, thereby overriding values with values that originated in the .ini file. In other words, the settings that are in the .ini file are used to replace the existing settings in the configuration string. At steps **1622** through **1662**, the available settings for UDI buttons (buttons are sometimes referred to as action, or click areas) that are defined in Table 1 are applied to the button. Thus, at step **1622** the "setting" is applied to the button to thereby associate Microsoft® Window Quick-launch or Favorites with the Zenu™ UDI for easy access by the user.

[0146] At a step **1624** a transparent setting can be applied to the button. Transparency allows buttons, and the like, to be visible while at the same time allowing the underlying image to be partially visible. Various known transparency techniques can be employed, as would become apparent to a person skilled in the computer graphics art. At step **1622** a resource bitmap is located, if so specified in the configuration string for the button in the theme file. If a bitmap is located, as determined at step **1628**, then the button bitmap is set to the located bitmap, at a step **1630**. Otherwise, the application resource is set to the UDI windows template file and the associated bitmap resource is searched for, as shown at step **1632**. If a template file bitmap resource is located, as determined at a step **1634**, flow proceeds to **1630**. Otherwise, the button is deleted as shown at a step **1636**. If the button is deleted, further buttons can be processed, as determined at a step **1638**. If so, a pointer to the next button is located, at a step **1640**, and flow proceeds to step **1606**.

[0147] After a bitmap is determined at either of step **1628** or step **1634**, it is applied at step **1630**. Next, a theme specific tool tip is set at a step **1644**. Then, at step **1646**, a "tweak" amount is set for button positioning. At a step **1548** an "auto repeat" feature is applied to the button if so desired. At step **1650**, a "check button" setting is applied to the button if so desired.

[0148] Next, at a step **1652**, it is determined whether a theme specific icon is specified for the button. If so, the theme specific icon is applied to the button, at step **1654**. Otherwise, flow proceeds to a step **1656** to determine whether there is a bitmap to use as an icon. If so, the icon bitmap is applied to the button at a step **1658**. Otherwise, flow proceeds to a step **1660**.