US 20120254733A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2012/0254733 A1

(43) Pub. Date: Oct. 4, 2012

Tucovic

(54) METHOD FOR USERS TO CREATE AND EDIT WEB PAGE LAYOUTS

(76) Inventor: **Aleksandar Tucovic**, Winnipeg (CA)

(21) Appl. No.: **13/073,432**

(22) Filed: **Mar. 28, 2011**

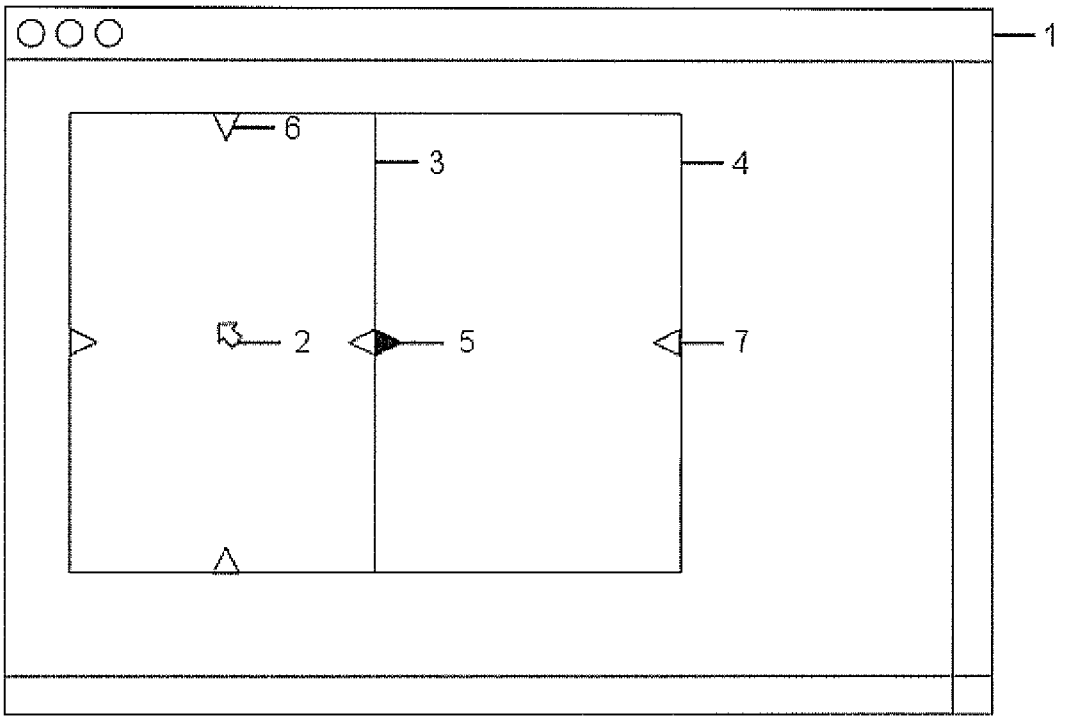
Publication Classification

(51) **Int. Cl.**
G06F 17/00 (2006.01)

(52) **U.S. Cl.** **715/243**

(57) **ABSTRACT**

A computer implemented method of editing a layout of areas on a page, for example for webpage design, includes displaying the layout in a user interface along with control elements operable via a user input device to manipulate user-variable attributes of the layout, the user-variable attributes including at least one of a number count of subdivisions in said layout, dimensions of said subdivisions in said layout, and border direction between each pair of adjacent subdivisions in said layout. Input via the user input device and the control elements is received to modify one or more of the user-variable attributes. Stored data representing the layout is updated based on the user input, and the display of the layout is updated based on the updated data to present a revised layout based on new values of the user-variable attributes.



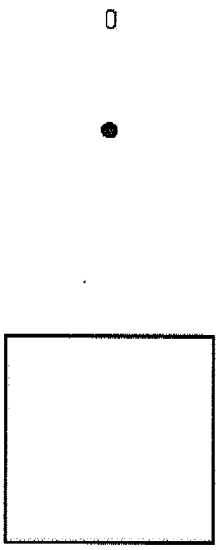


Figure 1

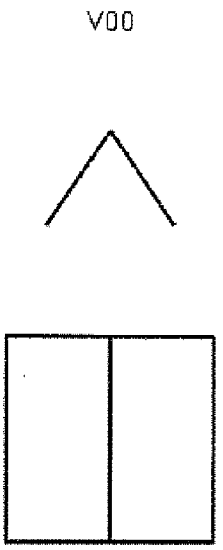


Figure 2

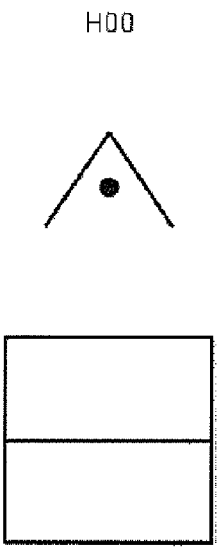


Figure 3

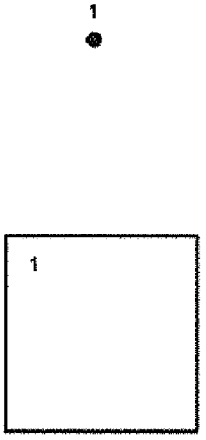


Figure 4

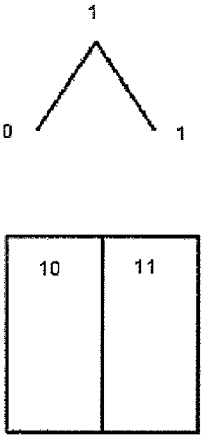


Figure 5

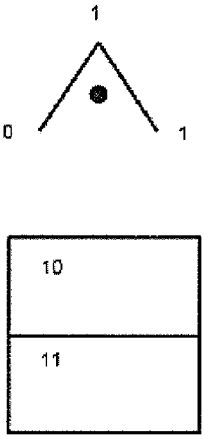


Figure 6

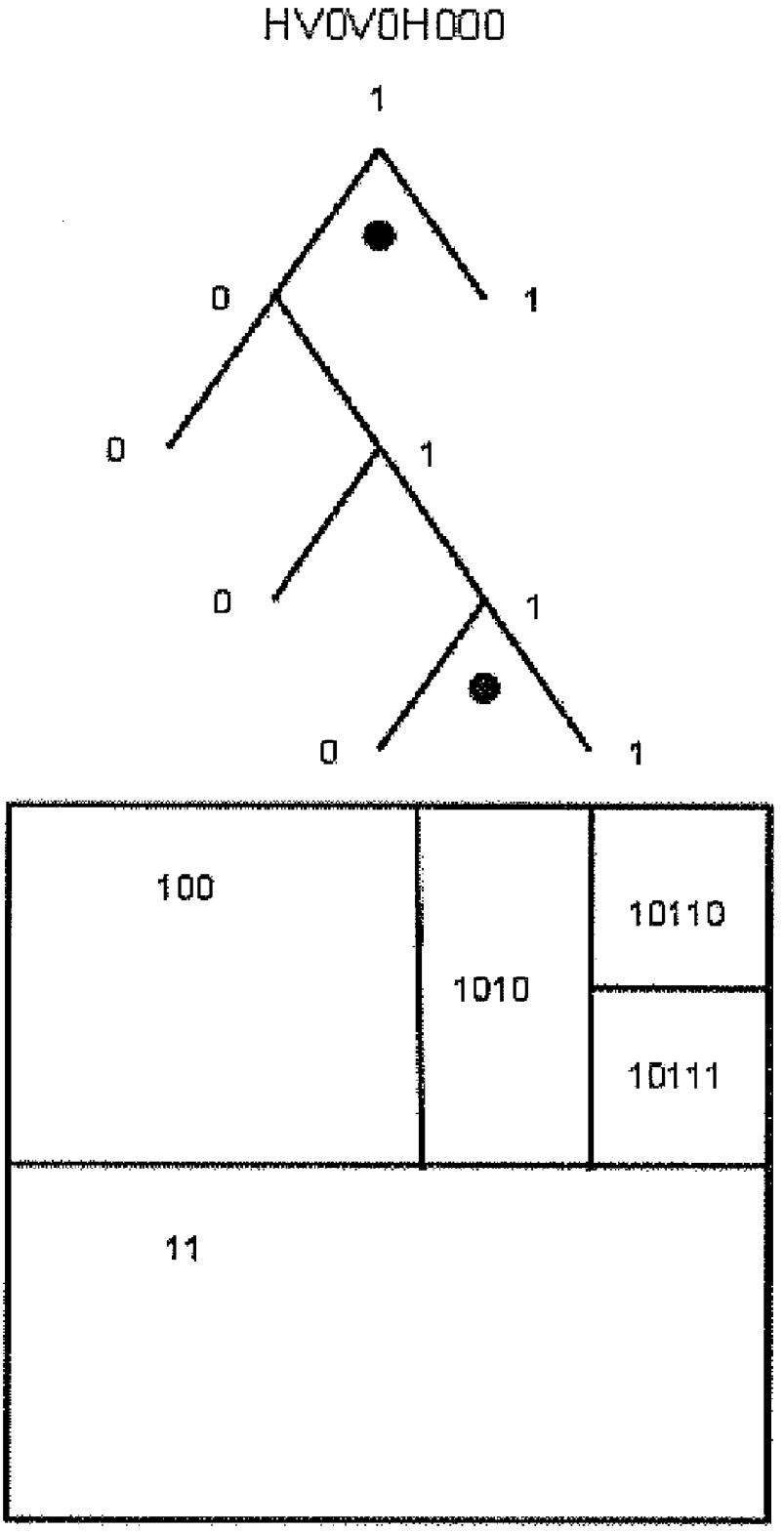


Figure 7

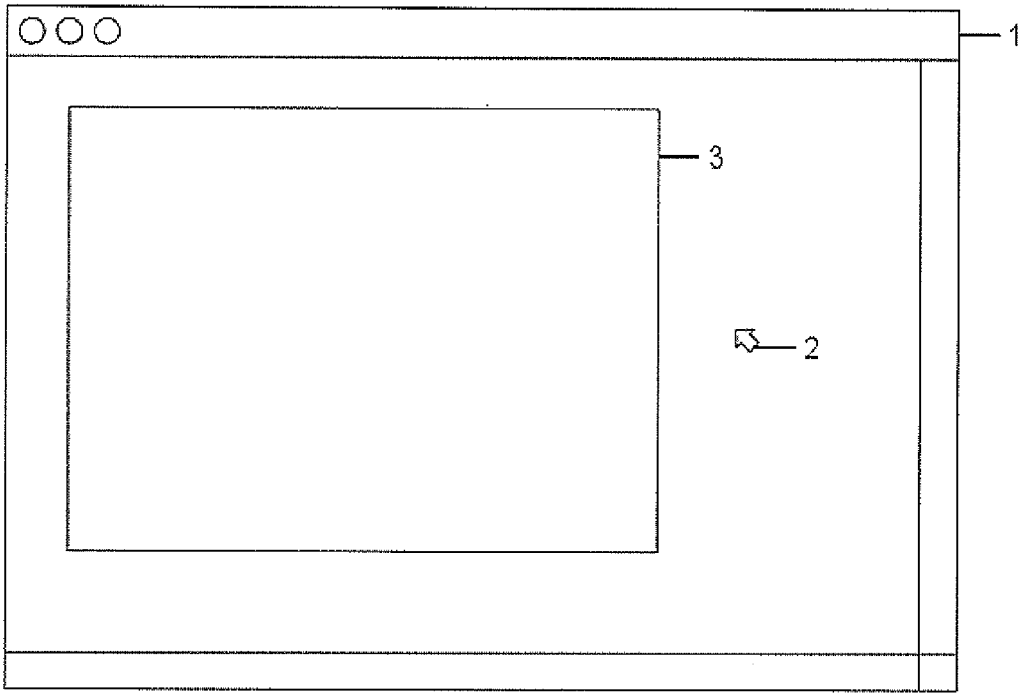


Figure 8

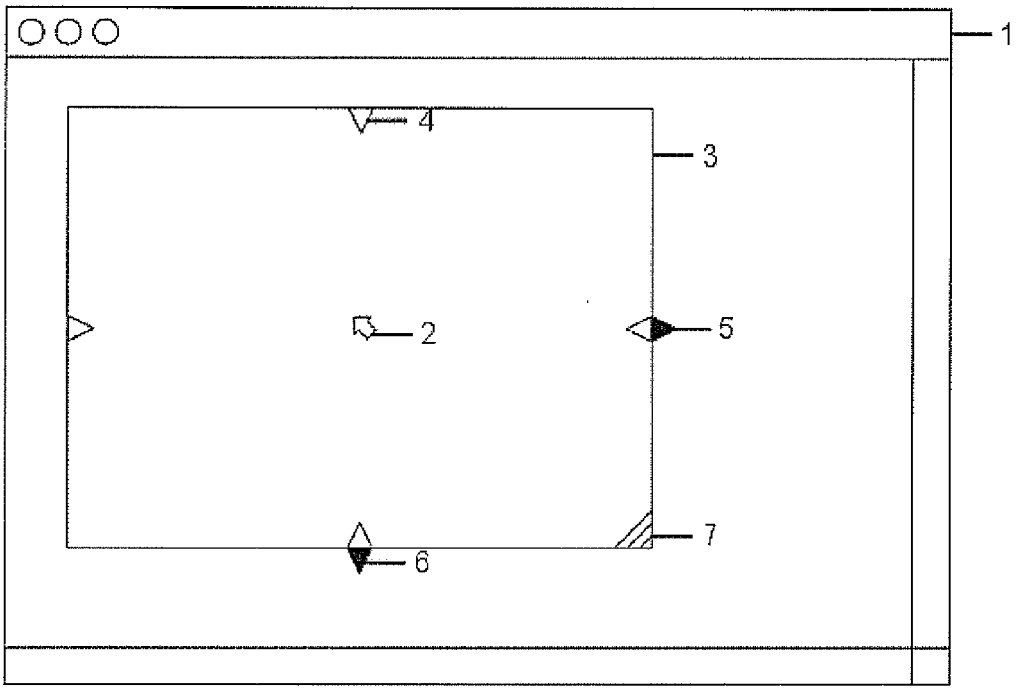


Figure 9

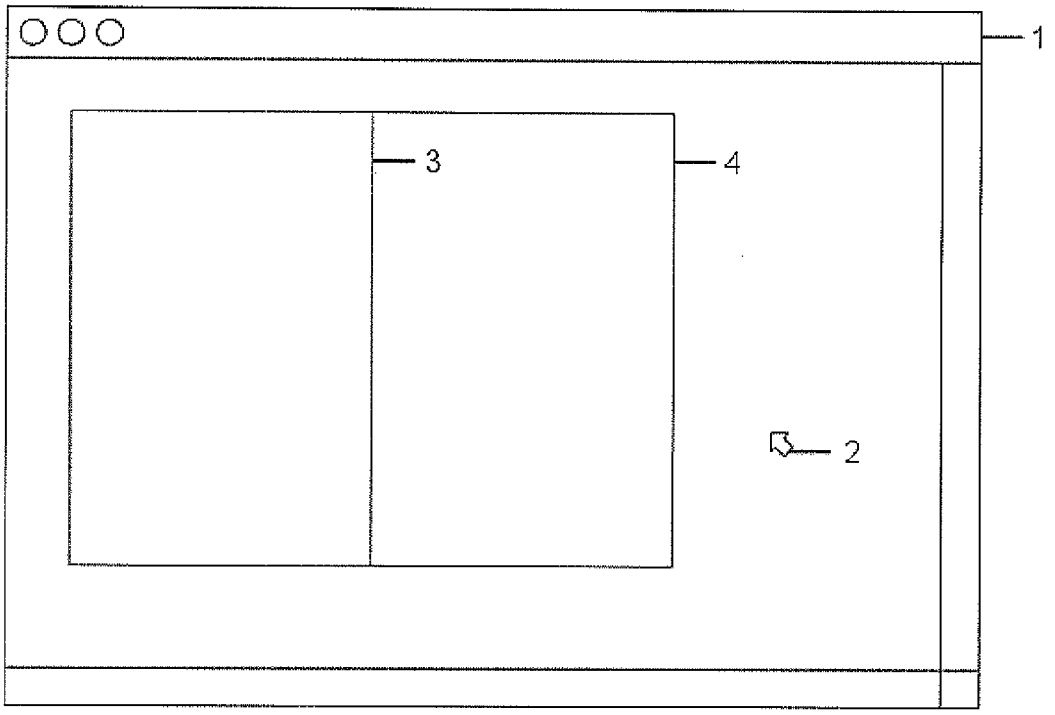


Figure 10

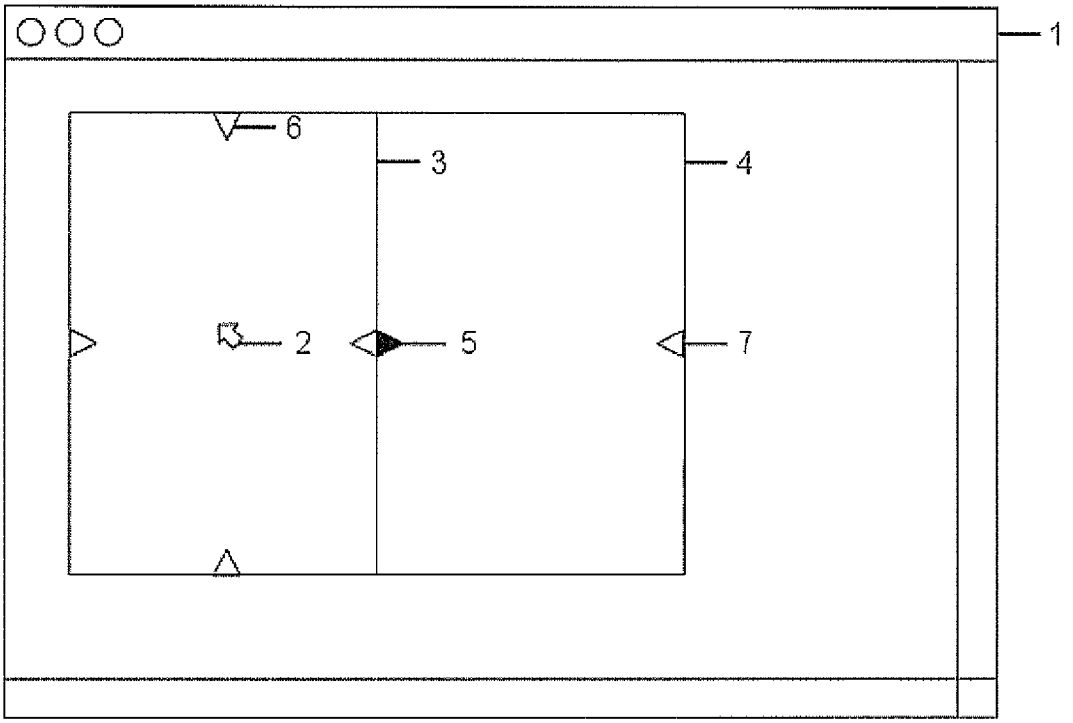


Figure 11

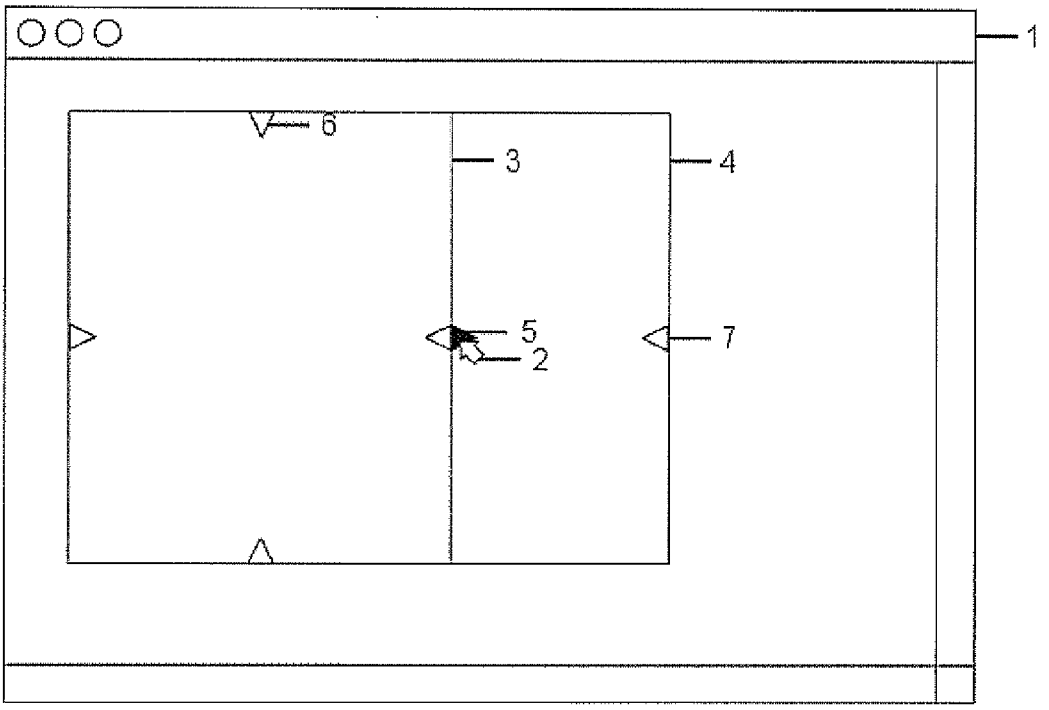


Figure 12

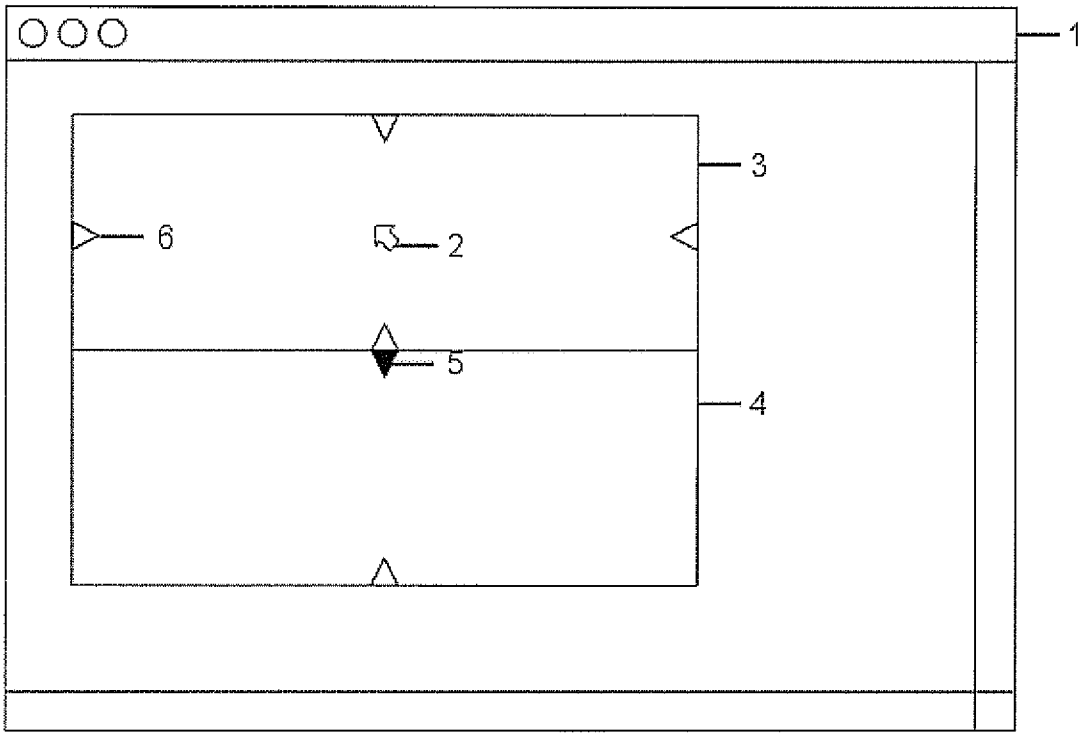


Figure 13

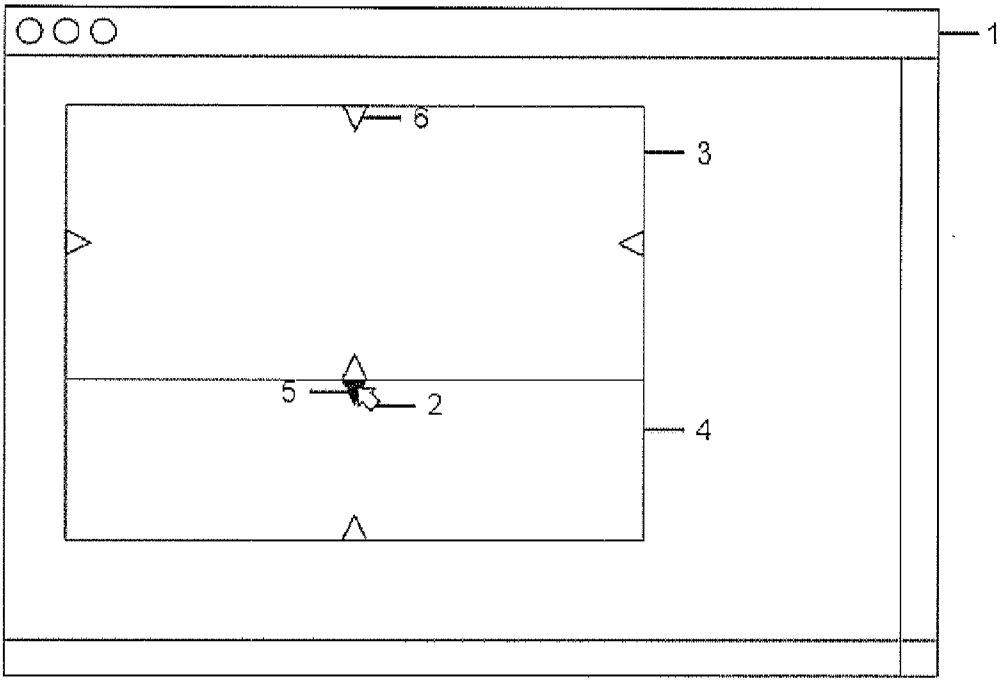


Figure 14 - Progressing from figure 13, resizing primary subdivision

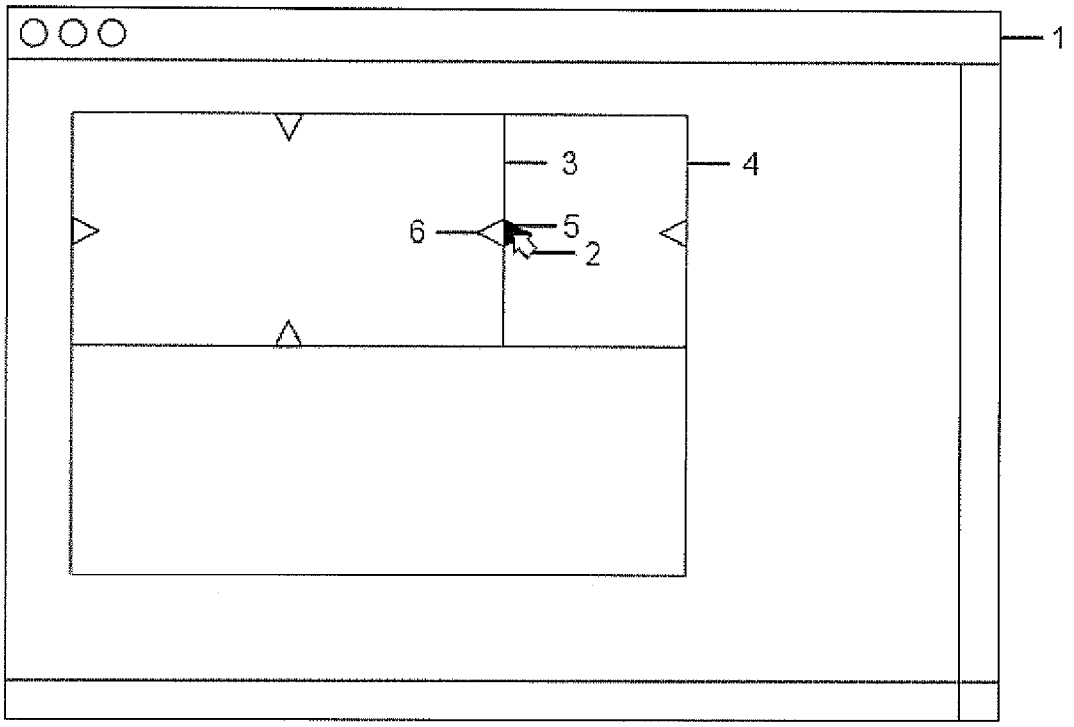


Figure 15 - Progressing from figure 13, left insertion from primary subdivision

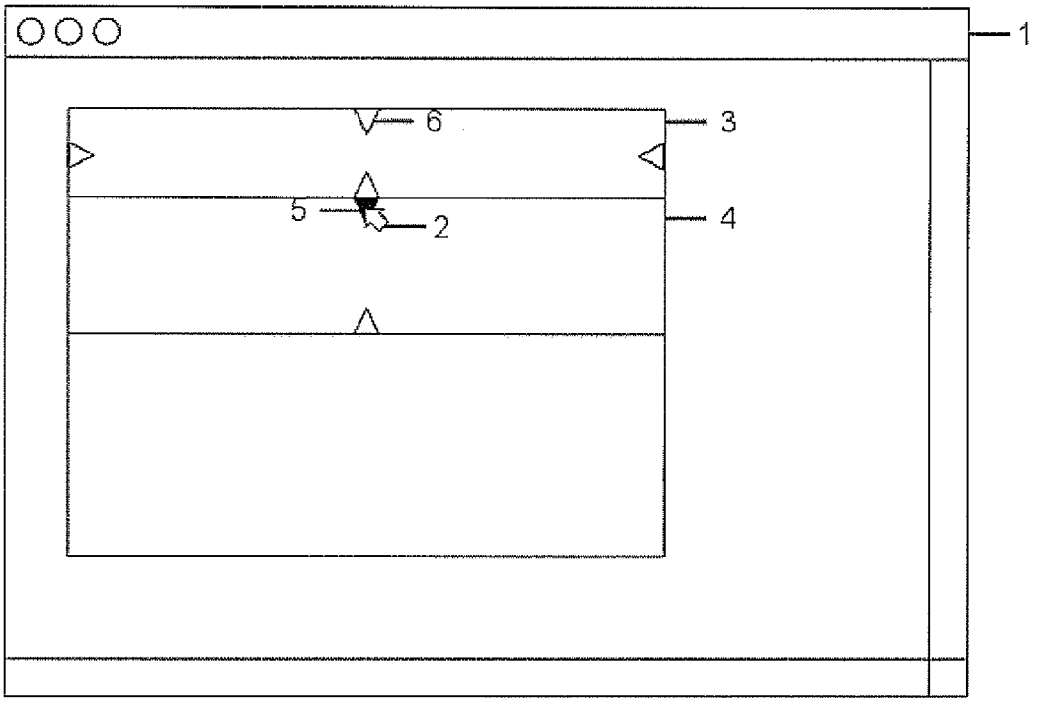


Figure 16 - Progressing from figure 13, top insertion from primary subdivision

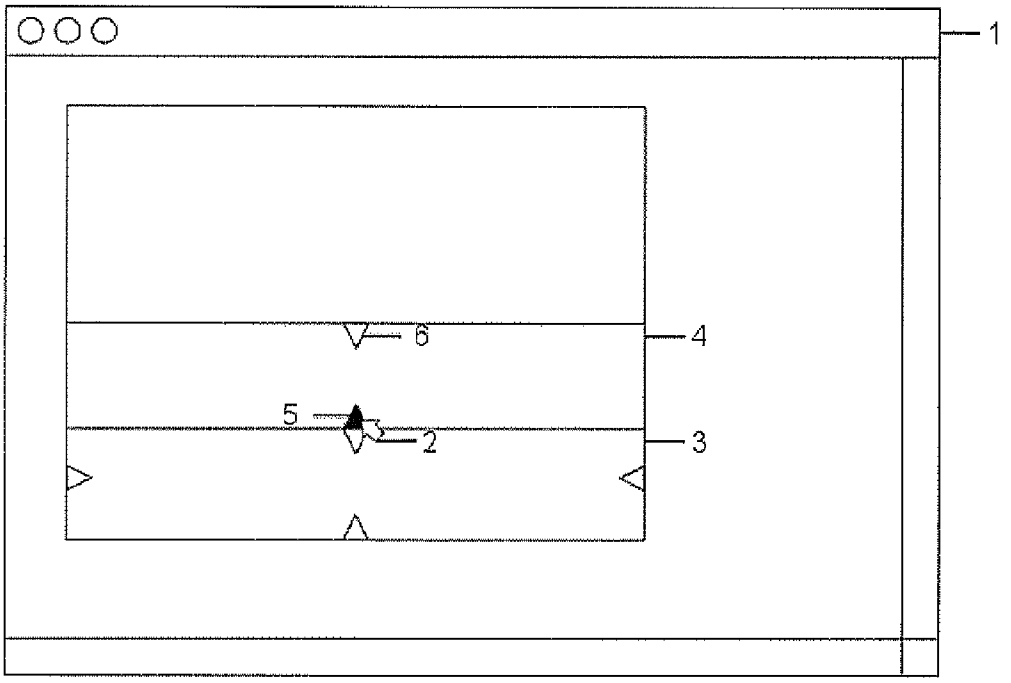


Figure 17 - Progressing from figure 13, bottom insertion from secondary subdivision

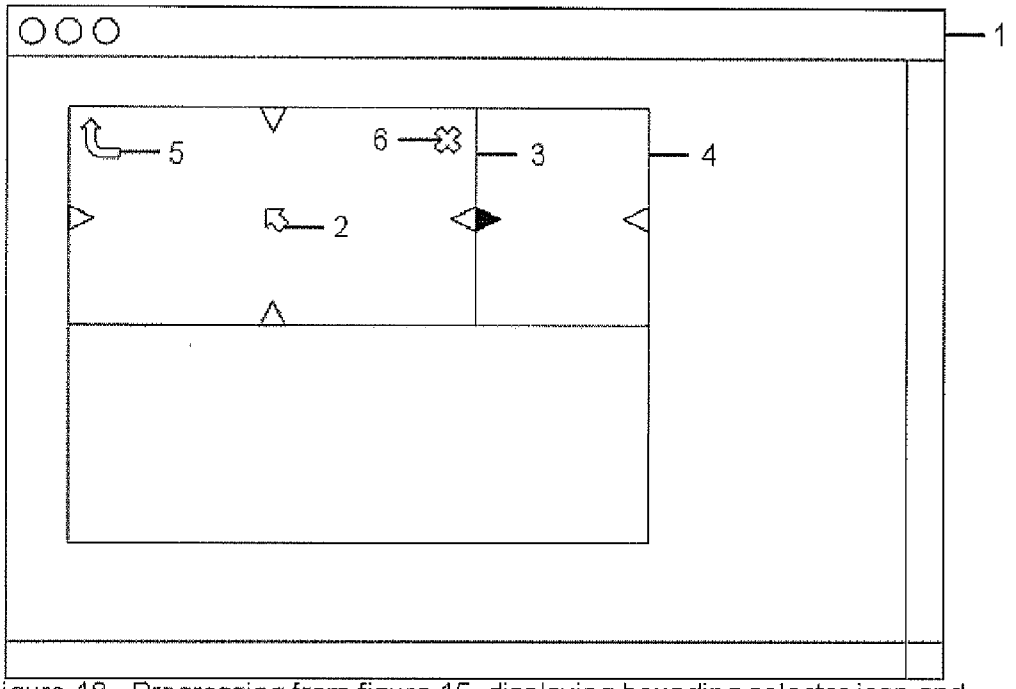


Figure 18 - Progressing from figure 15, displaying bounding selector icon and deletion icon

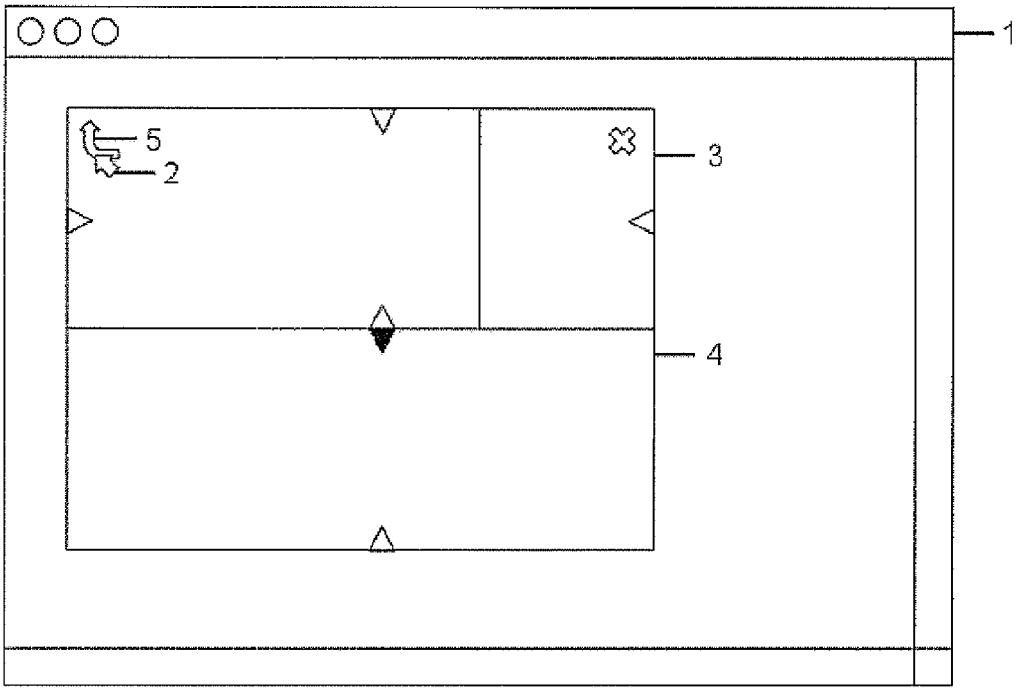


Figure 19 - Progressing from figure 18, primary subdivision's bounding division

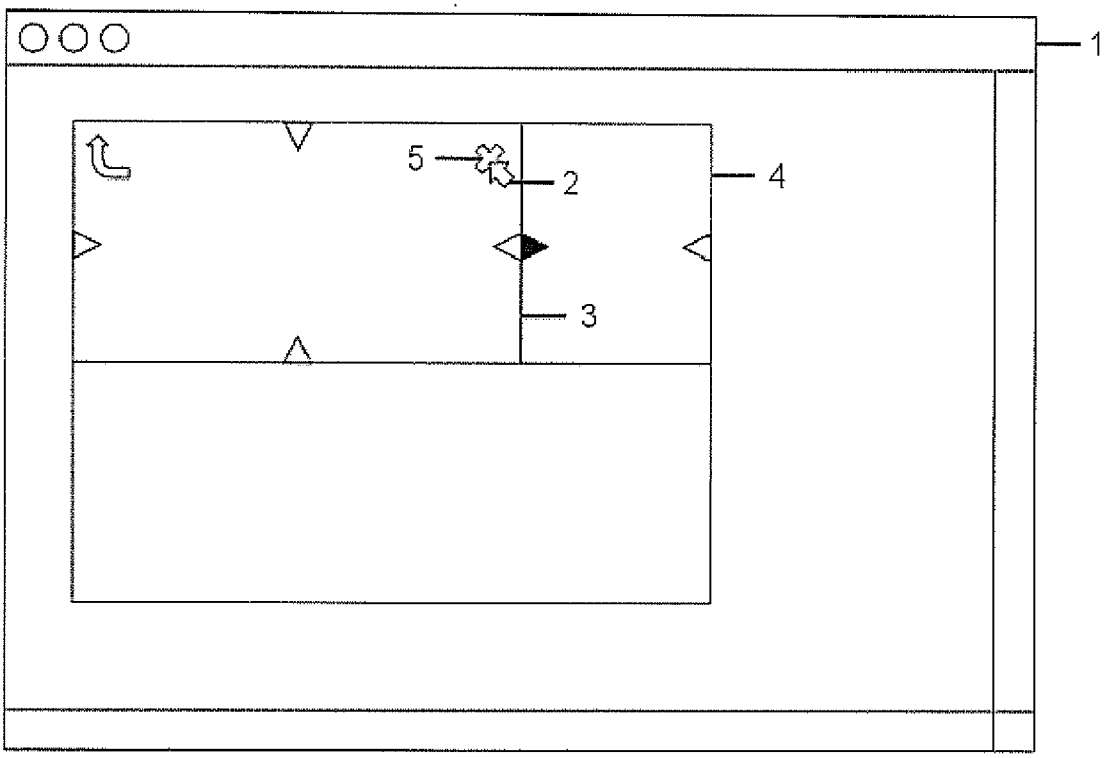


Figure 20 - Progressing to figure 13, deleting primary subdivision

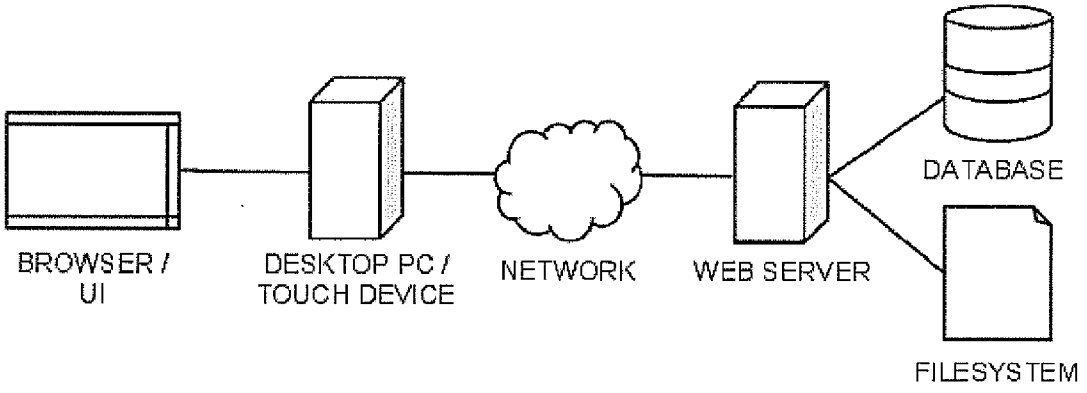


Figure 21 - Network Overview

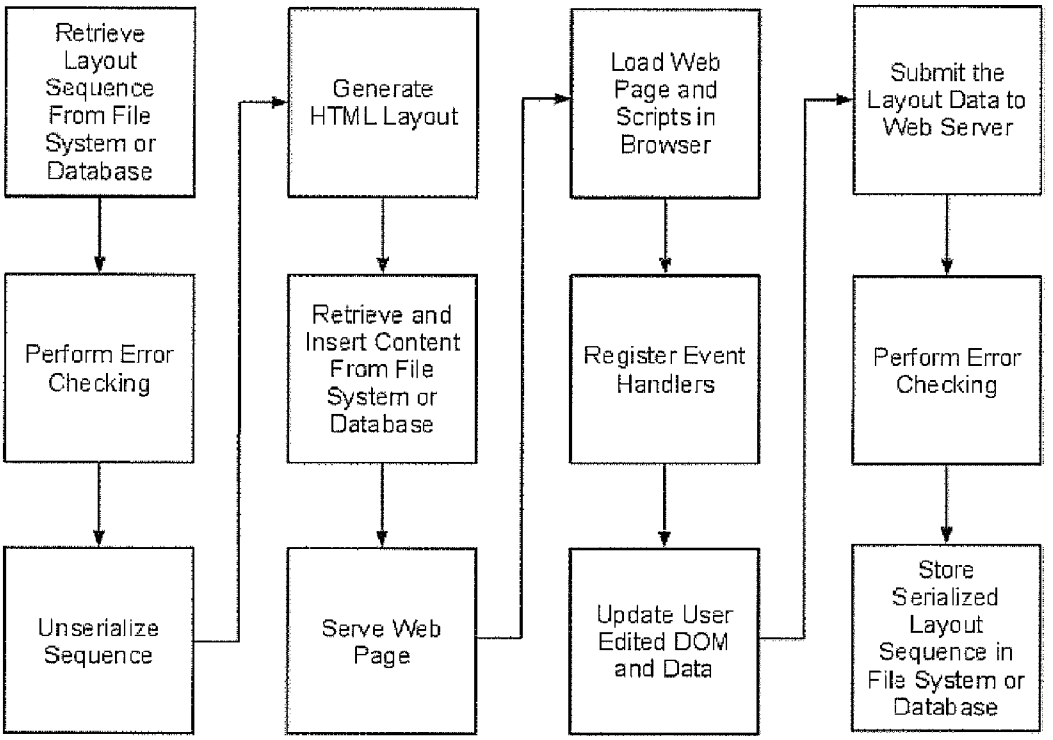


Figure 22 - Application Logic Sequence

METHOD FOR USERS TO CREATE AND EDIT WEB PAGE LAYOUTS

BACKGROUND OF THE INVENTION

[0001] This invention is a method for users to create and edit web page layouts in a consistent manner and offers a means for describing a web page layout of any level of complexity. This invention also allows for the possibility of storing and sharing layouts by means of a compact serialized sequence of characters representing the layout. This method also allows for templates to be embedded in other templates and for content to be included in the templates.

[0002] Website design is an evolving practice and there are modern standards for web page design. One of the more difficult things in web design is to define a layout in a standard way and edit it later especially if it is a complex layout. Even if standards were followed there are numerous approaches with some approaches favoured over others.

[0003] Prior art involves HTML documents coupled with CSS (cascading style sheets) to define the layout as written and stored in static text documents. The prior art in web page design includes misusing HTML TABLE tags that were meant to be used for tabular data but were used for layouts—TABLE tags does not lend itself well to layouts. Prior art is also accomplished via software that is used to design web page layouts in what is known as WYSIWYG (What You See Is What You Get) editors. WYSIWYG editors display an approximation of the final displayed layout. The invention allows for direct display and manipulation of the layout in modern web browsers. The invention encodes the layout in a compact format regardless of complexity of the layout. The compact format can then be stored, shared, used and reused in very sophisticated ways.

SUMMARY OF THE INVENTION

[0004] According to a first aspect of the invention there is provided a computer implemented method of editing a layout of areas on a page, comprising:

[0005] (a) in a user interface, displaying the layout along with control elements operable via a user input device to manipulate user-variable attributes of the layout, said user-variable attributes including at least one of a number count of subdivisions in said layout, dimensions of said subdivisions in said layout, and border direction between each pair of adjacent subdivisions in said layout;

[0006] (b) receiving input via the user input device and the control elements to modify one or more of said user-variable attributes;

[0007] (c) updating stored data representing the layout based on said user input; and

[0008] (d) updating the display of the layout in the user interface based on the updated data to present a revised layout based on new values of said one or more user-variable attributes.

[0009] Alternatively, step (b) may comprise receiving the input via a resizing element of the user interface associated with a particular one of a plurality of subdivisions of the layout, and step (c) comprises automatically updating size data associated at least one other subdivision based the user input.

[0010] Preferably the at least one other subdivision for which the size data is automatically adjusted includes two or more further subdivisions within the particular subdivision with which the resizing element is associated.

[0011] Preferably the size data of the further subdivisions is updated by splitting a size increase determined from the input for the particular subdivision among the further subdivisions based on relative sizing of said further subdivisions within said particular subdivision.

[0012] Preferably a larger portion of said increase is assigned to a larger one of said further divisions than to a smaller one of said further subdivisions.

[0013] Step (b) may comprise receiving the input via an insertion element of the user interface, and step (c) comprises automatically updating the data to reflect addition of a new subdivision inserted adjacent an existing subdivision and to automatically adjust size data of said existing subdivision and any previous subdivision based on insertion of the new subdivision.

[0014] Preferably the user input via the insertion element in step (b) includes sizing input for the new subdivision and step (c) comprises adding new size data for the new subdivision based on the sizing input.

[0015] Preferably the insertion element is a drag-and-drop element displayed at the respective side of the existing subdivision and draggable toward an opposite side thereof to create the new subdivision at said respective side.

[0016] Preferably four insertion elements are displayed in the user interface for the existing subdivision, one at each side of said existing subdivision.

[0017] Preferably drag-and-drop resize element is displayed for a secondary subdivision adjacent said existing subdivision at a side of the secondary subdivision adjacent said existing subdivision.

[0018] Step (b) may alternatively comprise receiving the input via a deletion element of the user interface associated with a particular one of a plurality of subdivisions of the layout, and step (c) comprises removing data associated with said particular subdivision from consideration in step (d) and automatically updating size data associated with one or more other subdivisions remaining in the layout to fill space previously occupied in said layout by said particular subdivision.

[0019] Preferably the stored data includes height and width data for the layout and, for each boundary between adjacent subdivisions, orientation data representing an orientation of said boundary within the layout and a single offset value that is used to position the boundary relative to a vertical or horizontal perimeter boundary of the layout.

[0020] Preferably the height and width data for the layout is used together with the single offset value for the boundary between the adjacent subdivisions to calculate a subdivision width and subdivision height of each of said adjacent subdivisions.

[0021] Preferably the user interface is displayed on a user device coupled to a server via a network connection, the server comprising memory containing the stored data representing the layout.

[0022] Preferably the user interface is displayed in a web browser of the user device.

[0023] According to a second aspect of the invention there is provided computer readable memory having recorded thereon computer executable statements and instructions that when executed by a computer perform a method of editing a layout of areas on a page, said method comprising:

[0024] (a) in a user interface, displaying the layout along with control elements operable via a user input device to

manipulate user-variable attributes of the layout, said user-variable attributes including at least one of a number count of subdivisions in said layout, dimensions of said subdivisions in said layout, and border direction between each pair of adjacent subdivisions in said layout;

[0025] (b) receiving input via the user input device and the control elements to modify one or more of said user-variable attributes;

[0026] (c) updating stored data representing the layout based on said user input; and

[0027] (d) updating the display of the layout in the user interface based on the updated data to present a revised layout based on new values of said one or more user-variable attributes.

[0028] According to a third aspect of the invention there is provided a computer implemented method of editing a layout of areas on a page, comprising:

[0029] displaying the layout in a user interface;

[0030] receiving input reflective of a user-selection of a subdivision within the layout;

[0031] displaying icons representative of physical bounds of the area within the layout;

[0032] facilitating selection of the icons representing the bounds of the subdivision within the layout or subdivision for either adjustment of a selected bound to a new position or insertion of a new subdivision into the layout; and

[0033] automatically adjusting one or more bounds of one or more other subdivisions within the layout in response to said adjustment or insertion to fill the layout with adjacent non-overlapping subdivisions.

[0034] Preferably the method includes storing a serialized sequence of values representative of the layout of adjacent non-overlapping subdivisions.

[0035] According to a fourth aspect of the invention there is provided a method of displaying a page layout comprising:

[0036] receiving serialized data representative of the page layout;

[0037] parsing the serialized to extrapolate data on the page layout, including data on subdivisions contained within an overall area of the page layout; and

[0038] using the extrapolated data to present an on-screen display of the page layout.

[0039] Preferably the data on the subdivisions includes orientation data reflecting orientations of boundaries between adjacent subdivisions.

[0040] Preferably a format of the serialized data employs a three position representation of a pair of adjacent subdivisions, in which a first position contains an orientation indicator reflecting the orientation of the boundary between the adjacent subdivisions, and second and third positions contain content indicators respectively reflective of contents of said adjacent subdivisions.

[0041] Where either one of said adjacent subdivisions defines a boundary subdivision containing a pair of smaller adjacent subdivisions therein, preferably a respective three position representation of said pair of smaller adjacent subdivisions therein occupies a respective one of the second and third positions of the three position representation of the adjacent subdivisions defining the boundary subdivision.

[0042] Preferably the serialized data includes two possible characters for each orientation indicator, one representing a vertical orientation and another representing a horizontal orientation, for example, the character 'H' for horizontal and the character 'V' for vertical.

[0043] Preferably a single predetermined character is used in the second or third position of any three position representation for which a respective one of the adjacent subdivisions is empty of any further smaller subdivisions therein, for example using the character '0'.

[0044] Preferably the serialized data includes a string containing separated dimensional values.

[0045] Preferably a first two dimensional values of the string represent an overall height and width of the layout and additional dimensional values of the string after the first two dimensional values correspond to the subdivisions within the layout and are sequenced in matching order with the orientation indicators of said subdivisions in the serialized data.

[0046] Preferably the additional dimensional values consist of only a single dimensional value for each pair of adjacent subdivisions, the method including using said single dimensional value as an offset relative to one dimension of a bounding area containing said pair of adjacent subdivisions to determine a first dimension of each of said adjacent subdivisions and inheriting the other dimension of the bounding area as a second dimension of each of said adjacent subdivisions.

BRIEF DESCRIPTION OF THE DRAWINGS

[0047] In the accompanying drawings, which illustrate exemplary embodiments of the present invention:

[0048] FIG. **1** is a schematic illustration of a empty rectangular layout and a graphical and string representation of same.

[0049] FIG. **2** is a schematic illustration and graphical and string representation of the layout of FIG. **1** after a single vertical division thereof to form two vertically oriented subdivisions.

[0050] FIG. **3** is a schematic illustration and graphical and string representation of the layout of FIG. **1** after a single horizontal division thereof to form two horizontally oriented subdivisions.

[0051] FIG. **4** is a modification of FIG. **1**, showing a unique identifier of the undivided area bound by the empty layout.

[0052] FIG. **5** is a modification of FIG. **2**, showing unique identifiers of the different areas bound by the two vertical subdivisions.

[0053] FIG. **6** is a modification of FIG. **3**, showing unique identifiers of the different areas bound by the two vertical subdivisions.

[0054] FIG. **7** is a schematic illustration and graphical and string representation of the layout of FIG. **1** after a series of divisions thereof to form a more complicated layout having five subdivisions, each labelled with its own unique identifier in the illustration.

[0055] FIG. **8** is a schematic screenshot illustration showing a user interface displaying the layout of FIG. **1** within a web browser.

[0056] FIG. **9** is a schematic screenshot illustration showing the display of user-operable subdivision insert handles on the layout of FIG. **8**.

[0057] FIG. **10** is a schematic screenshot illustration showing the user interface displaying the layout of FIG. **2**.

[0058] FIG. **11** is a schematic screenshot illustration showing the user interface and layout of FIG. **10** with insertion handles on a left or primary one of two vertical subdivisions

and a resize handle and insertion handle of a right or secondary one of the two vertical subdivisions thereof.

[0059] FIG. **12** is a schematic screenshot illustration showing the user interface of FIG. **11** after resizing of the vertical subdivisions via the resize handle.

[0060] FIG. **13** is a schematic screenshot illustration showing the user interface displaying the layout of FIG. **3** with insertion handles on a top or primary one of two horizontal subdivisions and a resize handle and insertion handle of a bottom or secondary one of the two horizontal subdivisions thereof.

[0061] FIG. **14** is a schematic screenshot illustration showing the user interface of FIG. **13** after resizing of the horizontal subdivisions via the resize handle.

[0062] FIG. **15** is a schematic screenshot illustration showing the user interface of FIG. **13** after insertion of a new vertical subdivision in the top horizontal subdivision.

[0063] FIG. **16** is a schematic screenshot illustration showing the user interface of FIG. **13** after insertion of a new horizontal subdivision over the previously primary subdivision.

[0064] FIG. **17** is a schematic screenshot illustration showing the user interface of FIG. **13** after insertion of a new horizontal subdivision below the secondary subdivision.

[0065] FIG. **18** is a schematic screenshot illustration of the layout of FIG. **15**, but additionally showing bounding selector and subdivision deletion icons respectively operable to select a bounding subdivision containing the currently selected subdivision primary subdivision or delete the currently selected primary subdivision.

[0066] FIG. **19** is a schematic screenshot illustration showing the user interface of FIG. **18** after selection of the bounding subdivision.

[0067] FIG. **20** is a schematic screenshot illustration demonstrating the use of the subdivision deletion icon of FIG. **18**, which reverts the layout of FIG. **18** back to that of FIG. **13**.

[0068] FIG. **21** is a flowchart illustration of a webpage layout creating and editing process of the present invention.

[0069] FIG. **22** is a schematic illustration of a network based system for webpage layout creation and editing.

DETAILED DESCRIPTION OF THE INVENTION

[0070] The invention is a compact description of a web page layout as well as a method for modifying it. A general formula defined by an algorithm expands the formula into common standards-based HTML and CSS which when served to the user's browser is displayed and can be directly modified. When the layout is saved the algorithm will revert the layout back into a compact encoding.

[0071] A layout for the purposes of the algorithm is a rectangular area that has zero or more non-overlapping rectangular subdivisions. The initial areas and subsequent subdivisions have zero margin, border and padding for simplicity although the algorithm can accommodate margins, borders and padding.

[0072] The algorithm works by describing a layout of an initial area with a width and height. An empty layout is described by the zero character '0' (FIG. **1**). An subdivided layout is 'X00' (FIGS. **2** and **3**). 'X' can represent either a horizontal or a vertical oriented subdivision represented respectively by 'H' or 'V'. When there are no more subdivisions it is represented by 'X' followed by '00'. Therefore 'X00' can represent either 'H00' or 'V00' as an initial area with only two subdivisions. Any '0' representing a respective rectangular area can be subdivided into two smaller areas by replacing it with 'X00' so that 'X00' becomes 'XX000' by replacing only the first '0' with 'X00', or 'X00' becomes 'X0X00' by replacing only the second '0', or 'X00' becomes 'XX00X00' by replacing each '0'. In 'H00', the first zero represents the top subdivision and the second zero represents the bottom subdivision. In 'V00', the first zero represents the left subdivision and the second zero represents the right subdivision.

[0073] For example, 'HV000' means that the first '0' of 'H00' is further divided into a pair of vertical subdivisions, while 'H0V00' means that the second '0' of 'H00' has been further divided into a pair of vertical subdivisions. 'HV00V00' means that both of the two horizontal subdivisions have each been vertically subdivided.

[0074] The string representation of the layout is the data that is stored either in a database system or file system.

[0075] Orientation is an important aspect of the algorithm. The algorithm codifies a layout using 'H' or 'V' instead of 'X'. The algorithm sequence can be graphed (FIGS. **1** to **7**). '0' is the initial empty (i.e. undivided) division (FIGS. **1** and **4**). The initial area '0' can be subdivided into either 'H00' (FIGS. **2** and **5**) or 'V00' (FIGS. **3** and **6**).

[0076] The layout of FIG. **7** and the corresponding string representation 'HV0V0H000' can be generated from the initial empty division '0' of FIG. **1** as follows:

[0077] 1. horizontally subdivide the initial area: 0→H00;

[0078] 2. vertically subdivide the top one (first zero) of the two horizontally divided areas resulting from the previous step: H00→HV000;

[0079] 3. vertically subdivide the right one (second zero) of the two vertically divided areas resulting from the previous step: HV000→HV0V000; and

[0080] 4. horizontally subdivide the right one (second zero) of two vertically divided areas resulting from the previous step: HV0V000→HV0V0H000.

[0081] The tree graph represents one aspect of the layout by representing the orientation of the subdivisions which in the figures is demonstrated by using a dot to mark the internal nodes of the tree that mark horizontal subdivisions. The lack of a dot at an internal node represents a vertical subdivision. The algorithm processes the graphed sequence in a top-down and left-right direction. Another aspect of the layout is represented by a binary value (i.e. either a 'zero' or 'one') at each node in the graph. By reading from the top and reading left-side first off the graph and appending the values for each path from the root node to a different leaf node, a unique identifier can be specified for each distinct subdivision as a string of binary digits. The unique identifiers are used to generate the web page layout for display in the user interface, and the unique identifiers for illustrated exemplary layouts are shown in association with their respective subdivisions in FIGS. **4** to **7**.

[0082] The algorithm also calculates the dimensions of the entire layout using the initial area's width and height. The layout sequence stores the layout's initial height and width but also stores one dimension for each 'X' in the sequence. The stored dimension is an offset dimension that marks a distance at which the boundary line of adjacent subdivisions should be spaced from a respective boundary of the overall layout. As the algorithm progresses through the layout sequence, the top or left one of two adjacent subdivisions has its dimensions calculated by inheriting the width of the area bounding the two adjacent subdivisions and applying the

offset height if they are horizontally divided CHI or by inheriting the offset width and the bounding area's height if they are vertically divided ('V'). The bottom or right subdivision's width in the case of 'H', or height in the case of 'V', is calculated by finding the difference in the respective dimension of the bounding area and the top or left subdivision, and the remaining dimension is inherited from the bounding division. This way the width and height of each subdivision is calculated by the algorithm as it progresses along the sequence using the initial layout's dimensions and 'X's offset dimensions. The amount of data required to be stored with only the offset tends towards a quarter compared to storing the height and width of each bounding division and each subdivision.

[0083] The dimensions are simply listed listed in order starting with the initial layout's width and height followed by a series of offset dimensions. Given a layout (FIG. **7**) with an initial width of 100 and height of 100, the offset for each 'X' would be represented, for example, by:

[0084] 100, 100, 53, 50, 27, 25

[0085] The compact format of the given layout (FIG. **7**) is stored in the general format:

[0086] HV0V0H000, 100, 100, 53, 50, 27, 25

[0087] The process for determining the layout begins by defining the dimensions of the initial area. The initial area's width and height are the first pair of stored dimensions. Following the definition of the initial area the sequence is parsed one character at a time. In the above example, 'H' is parsed first which determines that a top and bottom subdivision exists. When a 'X' value is parsed, then a stored dimension is parsed as well, which determines the subdivision dimensions. In this example, The third stored dimension of 53 is parsed and used to calculate the subdivision dimensions. The top and bottom subdivisions each inherit the initial area's width of 100. The top subdivision height is 53. The bottom subdivision height is the initial area's height minus the top subdivision (or stored dimension) height resulting in 100−53=47. The top subdivision dimensions are 100 pixels wide by 53 pixels high, and the bottom subdivision dimensions are 100 pixels wide by 47 pixels high. The two subdivisions are contained within the bounding division, defined in this example by the empty (i.e. undivided) initial layout.

[0088] Parsing follows this procedure:

1. START
2. Parse the first pair of stored dimensions and define the initial area's dimensions.
3. Parse the first character of the stored sequence
   a. If the first character is '0' then the parsing procedure skips to step (11)
   b. If the first character is 'X' then the parsing procedure continues to step to (4)
4. Last read 'X' is the bounding division
5. Parse the next dimension to define the subdivisions' dimensions based on the orientation in (4)
6. Parse the next character of the stored sequence
   a. If the character is '0' then the parsing is complete for the first subdivision and the parsing procedure skips to step (7).
   b. If the character is a 'X' then return to step (4) noting the 'X' as 'pending first subdivision'
7. Parse the next character of the stored sequence
   a. If the character is '0' then the parsing is complete for the second subdivision and parsing ends for the bounding division in (4), and the parsing procedure continues to step (8)
   b. If the character is a 'X' then return to step (4) noting the 'X' as 'pending second subdivision'
8. If there are no 'X's noted as 'pending subdivisions' then skip to step (11)
9. If parsing had noted the previous 'X' as 'pending first subdivision' then remove the 'X's pending status and continue to (7)
10. If parsing had noted the previous 'X' as 'pending second subdivision' then remove the 'X's pending status and continue to (8)
11. END

[0089] The algorithm allows for error checking because the algorithm and the layout sequence follows consistent rules. Once the layout sequence is accessed from storage error checking is performed against the sequence: the orientation sequence has two checks and the dimensions have one check. The layout sequence is first checked using a simple checksum and if that error check passes the layout sequence is processed by checking that the layout sequence simplifies to a valid initial area of '0'. The error checking is also performed before a layout sequence is stored.

[0090] Once the layout sequence is accessed from storage and passes error checking the layout and content is generated to be displayed in a web browser using standard web page markup and styles, or other graphic standards and software. In the case of web pages the layout generator may generate a web page with HTML with embedded CSS and JavaScript components. The web page generation all occurs on the web server. The resulting web page layout is displayed in the web browser (FIG. **8**).

[0091] The generated HTML layout specifies a partial DOM (Document Object Model) tree using a DIV markup. The DOM tree represents the initial area and each existing subdivision of the layout. Each DIV has an attribute ID with a unique string identifier as generated by the algorithm. In addition each DIV will have an attribute CLASS with a non-unique classification name based on its orientation as identified by 'X' in the layout sequence. A set of JavaScript functions are included in the web page that depend on the layout and the unique ID attribute and general CLASS attribute of each DIV generated by the algorithm. The JavaScript functions are classified generally by event handlers, calculators, layout generator and layout analyzer.

[0092] When the web page is served to the client browser the event handlers are attached to each subdivision as well as handles that will represent the bounds of selected subdivisions (FIGS. **8** and **9**). Each subdivision responds to a user click or touch that generates a data representation of the layout including orientation and dimensions. Subdivisions that have responded to user clicks or touches, known as primary nodes, have a complementary subdivision known as a secondary node. Handles are placed on the primary and secondary nodes.

[0093] A primary node has four handles that are known as insertion handles and placed on each side of the primary node subdivision (FIG. **11**, Item **3**). A secondary node (FIG. **11**, Item **4**) has two handles where one is an insertion handle and the other is known as a resize handle. The secondary node handles are placed on two sides of the secondary node where one handle is placed closest to the primary node and is the resize handle (FIG. **11**, Item **5**); the other handle is placed furthest from the primary node and is an insertion handle (FIG. **11**, Item **7**).

[0094] When a subdivision is selected, a JavaScript method generates a data structure of the primary and secondary sub-

division layout. The structure parses the unique identifier of the subdivision and logically determines in a recursive manner if there are any existing layouts or subdivisions within it. The recursive method exhaustively generates the data structure that stores each unique identifier, its orientation and its dimensions as well as a unique data structure that stores, for each pair of adjacent subdivisions, a reference to the subdivision that has the greater share of the bounding area containing those adjacent subdivisions, an antecedent of the ratio of the subdivision dimensions based on orientation and a cumulative change of one pixel differences. This data structure created by the layout generator is used in the editing of the layout.

[0095] To resize a subdivision it is clicked or touched for it to become the primary node with its four insertion handles (FIG. **11**, Item **6**) and a secondary node identified by two handles—a resize handle (FIG. **11**, Item **5**) and an insertion handle (FIG. **11**, Item **7**). Clicking or touching and holding a resize handle will activate the event handler that tracks how much the mouse moves onscreen while the handle is held (FIG. **12**). The difference in position is simply added to the primary dimension and subtracted from the secondary dimension relative to the orientation of the primary and secondary nodes. This addition and subtraction occurs first in the data structure of the layout and then applied to the style property of the DIV node to reflect the size change onscreen. For example, movement of a resize handle of a secondary vertical subdivision tracks a horizontal (x-coordinate) component of the movement in order to horizontally reposition the handle-equipped vertical boundary between the subdivisions. On the other hand, movement of a resize handle of a secondary horizontal subdivision tracks a vertical (y-coordinate) component of the movement in order to vertically reposition the handle-equipped horizontal boundary between the subdivisions.

[0096] A special case exists when the initial area is selected as the primary node (FIG. **9**). There is no secondary node when an initial area is selected. Insertion handles are placed as per selection of any subdivision however additional resize handles are placed. A resize handle is placed on the right side (FIG. **9**, Item **5**) and one is placed on the bottom side (FIG. **9**, Item **6**). A resize handle is placed in the corner of the initial area to indicate that both width and height can be resized concurrently (FIG. **9**, Item **7**). Resizing the initial area follows the same rules as resizing any subdivision.

[0097] There are two ways in which the difference is split: crisp logic and fuzzy logic. The edge case of a difference of one pixel being split between two subdivisions is taken care of first. A cumulative value is tracked against the larger subdivision. A one pixel change occurring via the resize handle between the two subdivisions in question increments or decrements the cumulative change by one. The ratio is calculated as the larger subdivision's dimension relative to the smaller subdivision's dimension, and it is updated as the dimension changes. Since a change of one pixel cannot be split between two subdivisions it is transferred to the subdivision that has the largest share of the bounding division's dimension, so long as the cumulative single-pixel change count is less than or equal to the ratio of the larger subdivision's dimension to the smaller subdivision's dimension. However, if after any given single-pixel change input from the resize handle the absolute value of the cumulative change is greater than the ratio, then the one pixel is added to the smaller subdivision's dimension instead and the cumulative change is reset to zero.

[0098] For example, consider two adjacent horizontal subdivisions with one height Y_1 and the other height Y_2. If Y_1 is greater than Y_2 then the ratio is Y_1/Y_2 otherwise it is Y_2/Y_1. When the ratio is exactly 1 (i.e. $Y_1=Y_2$) then the top subdivision (or left subdivision if the subdivisions are vertically divided) is represented by the fractional ratio, otherwise it is the division with greater dimension. A detected one-pixel increase in the bounding division size increments the initially zero cumulative change value up to one. If the cumulative change is less than the value of the ratio, the single-pixel increase is added to the subdivision represented by the antecedent (i.e. the larger subdivision). The ratio is recalculated with the new dimensions. Under further single-pixel increases of the bounding division, the cumulative change count is incremented by one. If at any time the cumulative change is greater than the ratio the one-pixel change is instead added to the smaller subdivision and the cumulative change value is reset to zero.

[0099] The second case is a fuzzy logic rule that applies when the difference is greater than 1 but less than some reasonable but adjustable limit. Fuzzy logic rules apply a set of rules to varying degrees to find a fuzzy weighted average that specifies how much difference should be propagated to the subdivisions' dimensions. The fuzzy weighted difference may end up less than the actual change in dimension, this difference being a subsequent difference. The subsequent difference is then propagated to the subdivisions as a change in dimension of one pixel as many times as necessary to make up any difference in the fuzzy weighted difference with the actual change in dimension.

[0100] The third case occurs when the difference is more than the reasonable limit set for the fuzzy logic rules. A formula splits the difference between two subdivisions based on each subdivision's share of the bounding division's dimension, determining the change for each subdivision as:

(1+difference/bounding division's dimension)×subdivision's dimension

[0101] The above formula behaves well when the difference is greater than the reasonable limit however when the difference is less than the reasonable limit and greater than one the difference is split with a large bias toward one subdivision therefore a fuzzy weighted difference is required to give a balanced split in difference.

[0102] A special case exists in that another distinct layout can be placed inside an initial area or a subdivision. When a bounding division is resized the difference is applied to the appropriate dimension of the initial area of the distinct layout. The difference is then propagated within the distinct layout according to the resize rules above.

[0103] The user continues to modify the layout by selecting any subdivision and the application creates the underlying data structure and updates the graphical layout. When the user is ready they can save the layout's dimensions. The algorithm determines the underlying layout by starting with the initial area and recursively determining the dimensions and orientation of each subdivision. The data structure is then submitted to the server-side application which serializes the data structure and performs error checking. If the serialization passes error checking then the serialized data is stored in the database or file system otherwise the user is notified of an error.

[0104] A layout can also be modified by inserting or deleting subdivisions. In order to insert a subdivision a user must select a candidate subdivision that represents the ideal place-

ment beside the insertion (FIG. **13**). Once the primary node is selected then the user needs to select a handle and adjust the position of the handle toward the opposite handle in the subdivision, thus defining a new subdivision at a location between the side of the existing subdivision from which the handle was dragged and the opposing side of that subdivision thereby splitting that subdivision into two further subdivisions.

[0105] A new subdivision can be inserted on any side of a primary subdivision by selecting a subdivision and then selecting any of the appropriate handles of it or selecting the insertion handle of the secondary subdivision. A user inserts a new subdivision to the left of a primary subdivision by selecting and holding the left insertion handle and then moving towards the right handle (FIG. **15**). A user inserts a new subdivision on top of a primary subdivision by selecting and holding the top insertion handle and then moving down toward the bottom handle (FIG. **16**). Inserting a new subdivision to the right or bottom of the primary subdivision is done in an analogous manner as to the left and top sides. In visual terms, the position to which the respective side or boundary was dragged by the insertion handle defines the position of the new dividing boundary created through this action to split the handle-dragged subdivision into two subdivisions.

[0106] By selecting a primary subdivision, the secondary subdivision is also displayed with its resize handle and its insertion handle. A new subdivision can also be inserted on the one side of secondary subdivision indicated by the insertion handle furthest away from primary subdivision (FIG. **17**) by dragging the insert handle at this outer side of the secondary subdivision toward the primary subdivision.

[0107] Selecting a subdivision and inserting new subdivisions will actually modify the data structure, modify the DOM tree and the insertion functionality will revert to a resize functionality. Initially, a subdivision is selected as a primary subdivision, but once the action of inserting a new subdivision is committed via an insertion handle of that primary subdivision, then the newly inserted subdivision becomes the primary subdivision (FIGS. **13**, **15**, **16**) and the remainder of the former primary subdivision not occupied by the new subdivision becomes the secondary subdivision. On the other hand, when the insertion is committed on the secondary subdivision the remaining portion of that secondary subdivision not occupied by the new subdivision remains as the secondary subdivision, and the newly inserted subdivision is the primary subdivision (FIGS. **13** and **17**). The insertion handle that was just used to create the new subdivision is now associated with the newly created subdivision boundary, and changes immediately to a resize handle of the secondary subdivision for the remainder of the user action.

[0108] There are two special cases that are additionally handled and they are separate actions of deleting a subdivision and selecting a bounding division (i.e. how to select an area that contains subdivisons within in at, as merely clicking a point somewhere within the area will instead select the area's particular subdivision in which the point resides).

[0109] A layout will display only subdivisions and only terminal divisions (i.e. subdivisions with no further subdivisions within them) will respond to click or touch events. Therefore, in order to select a bounding division a primary subdivision needs to be selected. Selecting a primary subdivision, in addition to displaying the aforementioned resize and insertion handles, also displays a deletion icon (FIG. **18**, Item **5**) and a bounding division icon (FIG. **18**, Item **6**). By selecting a bounding division icon it triggers the bounding division containing the previously selected subdivision to be selected as the new primary subdivision, and accordingly causes the handles, deletion and bounding division icon to be displayed in association with this bounding division instead of the previously selected subdivision contained therein (FIG. **19**).

[0110] Deleting a subdivision is done by clicking or touching the deletion icon in the primary subdivision, and preferably confirming the deletion (FIG. **20**) through some further action, for example through a dialog box. By deleting a primary subdivision any initial area and any and all subdivisions or content contained within it will also be deleted. The secondary subdivision will also be deleted but then any initial area and any or all subdivisions and content contained within it will become the initial area or subdivisions and content of the bounding division and will expand to the dimensions of the bounding division using the resize algorithm.

[0111] Although the preferred embodiment is described in terms of webpage layouts, it will be appreciated that similar division of an onscreen page or layout may be useful in other areas. Also, while the described embodiment uses employs a webserver to interact with users over the Internet via a web browser-displayed user interface, it will be appreciated that other embodiments may operate in other networked environments or operate as a locally installed software application on a workstation, desktop computer, laptop, tablet, PDA, mobile/cellular smartphone, etc. to generate customized layouts.

[0112] Since various modifications can be made in my invention as herein above described, and many apparently widely different embodiments of same made within the spirit and scope of the claims without department from such spirit and scope, it is intended that all matter contained in the accompanying specification shall be interpreted as illustrative only and not in a limiting sense.

1. A computer implemented method of editing a layout of areas on a page, comprising:

(a) in a user interface, displaying the layout along with control elements operable via a user input device to manipulate user-variable attributes of the layout, said user-variable attributes including at least one of a number count of subdivisions in said layout, dimensions of said subdivisions in said layout, and border direction between each pair of adjacent subdivisions in said layout;

(b) receiving input via the user input device and the control elements to modify one or more of said user-variable attributes;

(c) updating stored data representing the layout based on said user input; and

(d) updating the display of the layout in the user interface based on the updated data to present a revised layout based on new values of said one or more user-variable attributes.

2. The method of claim **1** wherein step (b) comprises receiving the input via a resizing element of the user interface associated with a particular one of a plurality of subdivisions of the layout, and step (c) comprises automatically updating size data associated at least one other subdivision based the user input.

3. The method of claim **2** wherein the at least one other subdivision for which the size data is automatically adjusted

includes two or more further subdivisions within the particular subdivision with which the resizing element is associated.

4. The method of claim **3** wherein step (c) comprises updating the size data of the further subdivisions by splitting a size increase determined from the input for the particular subdivision among the further subdivisions based on relative sizing of said further subdivisions within said particular subdivision.

5. The method of claim **4** wherein step (c) comprises assigning a larger portion of said increase to a larger one of said further divisions than to a smaller one of said further subdivisions.

6. The method of claim **1** wherein step (b) comprises receiving the input via an insertion element of the user interface, and step (c) comprises automatically updating the data to reflect addition of a new subdivision inserted adjacent an existing subdivision and to automatically adjust size data of said existing subdivision and any previous subdivision based on insertion of the new subdivision.

7. The method of claim **6** wherein the user input via the insertion element in step (b) includes sizing input for the new subdivision and step (c) comprises adding new size data for the new subdivision based on the sizing input.

8. The method of claim **7** wherein the insertion element is a drag-and-drop element displayed at the respective side of the existing subdivision and draggable toward an opposite side thereof to create the new subdivision at said respective side.

9. The method of claim **8** comprising displaying four insertion elements in the user interface for the existing subdivision, one at each side of said existing subdivision.

10. The method of claim **9** comprising, for a secondary subdivision adjacent said existing subdivision, displaying a drag-and-drop resize element for said secondary subdivision at a side thereof adjacent said existing subdivision.

11. The method of claim **1** wherein step (b) comprises receiving the input via a deletion element of the user interface associated with a particular one of a plurality of subdivisions of the layout, and step (c) comprises removing data associated with said particular subdivision from consideration in step (d) and automatically updating size data associated with one or more other subdivisions remaining in the layout to fill space previously occupied in said layout by said particular subdivision.

12. The method of claim **1** wherein the stored data includes height and width data for the layout and, for each boundary between adjacent subdivisions, orientation data representing an orientation of said boundary within the layout and a single offset value that is used to position the boundary relative to a vertical or horizontal perimeter boundary of the layout.

13. The method of claim **12** comprising using the height and width data for the layout together with the single offset value for the boundary between the adjacent subdivisions to calculate a subdivision width and subdivision height of each of said adjacent subdivisions.

14. The method of claim **1** wherein the user interface is displayed on a user device coupled to a server via a network connection, the server comprising memory containing the stored data representing the layout.

15. The method of claim **14** wherein the user interface is displayed in a web browser of the user device.

16. A computer implemented method of editing a layout of areas on a page, comprising:

displaying the layout in a user interface;

receiving input reflective of a user-selection of a subdivision within the layout;

displaying icons representative of physical bounds of the area within the layout;

facilitating selection of the icons representing the bounds of the subdivision within the layout or subdivision for either adjustment of a selected bound to a new position or insertion of a new subdivision into the layout; and

automatically adjusting one or more bounds of one or more other subdivisions within the layout in response to said adjustment or insertion to fill the layout with adjacent non-overlapping subdivisions.

17. The method of claim **16** comprising storing a serialized sequence of values representative of the layout of adjacent non-overlapping subdivisions.

18. A method of displaying a page layout comprising:

receiving serialized data representative of the page layout;

parsing the serialized to extrapolate data on the page layout, including data on subdivisions contained within an overall area of the page layout; and

using the extrapolated data to present an on-screen display of the page layout.

19. The method of claim **18** wherein:

a format of the serialized data employs a three position representation of a pair of adjacent subdivisions, in which a first position contains an orientation indicator reflecting the orientation of the boundary between the adjacent subdivisions, and second and third positions contain content indicators respectively reflective of contents of said adjacent subdivisions; and

the serialized data includes separated dimensional values, of which a first two dimensional values represent an overall height and width of the layout and additional dimensional values after the first two dimensional values correspond to the subdivisions within the layout and are sequenced in matching order with the orientation indicators of said subdivisions in the serialized data.

20. The method of claim **19** wherein the additional dimensional values consist of only a single dimensional value for each pair of adjacent subdivisions, the method including using said single dimensional value as an offset relative to one dimension of a bounding area containing said pair of adjacent subdivisions to determine a first dimension of each of said adjacent subdivisions and inheriting the other dimension of the bounding area as a second dimension of each of said adjacent subdivisions.

* * * * *