EasyChair Preprint

№ 10801

A Toolkit for Automatically Generating and Modifying VR Hierarchy Tile Menus

Xiyu Bao, Yulong Bian, Meng Qi, Yu Wang, Ran Liu, Wei Gai, Juan Liu, Hongqiu Luan and Chenglei Yang

August 29, 2023

CGI2018 manuscript No.
(will be inserted by the editor)

A Toolkit for Automatically Generating and Modifying VR Hierarchy Tile Menus

Abstract Current VR/AR system development studios lack a toolkit to automatically generate hierarchical tile menu layouts and menu prototypes for VR/AR devices without the need for user programming. This paper proposes a toolkit that automatically generates a hierarchy tile menu layout via a modified circular treemap algorithm and allows users to interactively arrange and resize tiles to form various layouts with their preferences or needs via a circle packer method and then automatically generates a VR/AR menu prototype based on the outputted layout. Moreover, reprogramming is also not required each time when the hierarchy is modified, or the menu layout is redesigned. The user test shows that the proposed toolkit simplifies the creation of hierarchy tile menu layouts, improves the creation efficiency of users, and allows users to flexibly create hierarchical tile menu prototypes based on their design idea.

Keywords Tile menu · Toolkit · User Interface · Virtual Reality/Augmented Reality

1 Introduction

The rise of the Metaverse has increased the interest in Virtual Reality (VR) and Augmented Reality (AR), making VR/AR applications more promising. Interaction menu techniques in VR/AR applications have become a significant research topic [13]. While 3D menus have been explored [12,27,7], adaptive 2D menus are commonly used in VR/AR applications due to users' familiarity with controlling 2D menus [24].

Tile menus are widely used in VR/AR applications due to their efficient and flexible selection operations. They allow users to arrange and resize each tile in a grid format [5,36,6]. Common forms of tile menus in VR/AR include Palace menus (e.g., the main menu on HoloLens 2), Coverflow menus [3], Gallery menus (e.g., the library menu on Oculus Quest 2), and Drawer menus (e.g., the menu toolbar on Oculus Quest 2) [13].

Several applications and studies have been conducted on tile menus. Microsoft introduced tile menus in the Windows 8 start screen for launching programs and applications [11]. Live Tile Anywhere [1] is an app that allows users to create custom Live Tiles and add them as widgets on their desktops. In the VR/AR domain, gMenu system was proposed [16], which uses a scalable grid to create customizable tile menu layouts. However, existing VR/AR development platforms only support grid-based tile arrangements with rectangular shapes, lacking hierarchical relationships.

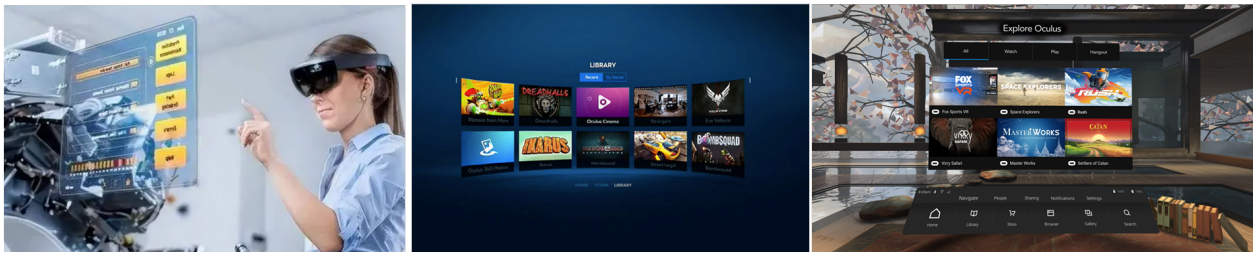
Prototyping hierarchical tile menus for VR/AR environments traditionally involves UI designers using tools like Adobe Illustrator to create menu layouts and programmers importing the designs into development platforms (e.g., Unity3D). Programmers also need to implement hierarchical relationships and menu view switching when users interact with the buttons.

To address these challenges, this paper proposes a toolkit for automatically generating hierarchical tile menu layouts and prototypes. The toolkit consists of two parts: Generator and Creator. The Generator generates concise layouts using a modified circular treemap algorithm, allowing interactive tile arrangement and resizing. The Creator automatically generates menu prototypes in Unity based on the layouts. User studies validate the effectiveness of the toolkit.

In summary, our contributions are as follows:

(1) We proposed two tools: Generator and Creator. The Generator can automatically generate hierarchical tile menu layouts from user input and allow users to interactively rearrange and resize tiles from various

Address(es) of author(s) should be given



(a)

(b)

Fig. 1: Comparison of control interfaces generated by our toolkit and the common AR/VR system control interfaces. (a) Common AR/VR tile menus. most of which are generated based on rectangular mesh, herein, the left is an A menu, the middle is a VR menu, and the right is a hierarchical menu: (b) Tile menus with various layouts generated by our toolkit, herein, the left is a tile menu generated by packing hexagon tiles, the middle is tile menus generated by packing triangle, rectangle and circle tiles in different shaped canvases, and the right is a hierarchical tile menu.

layouts in real time. The Creator can automatically prototype hierarchical tile menus based on the layout from the Generator in Microsoft HoloLens 2.

(2) We proposed a method to automatically pack regular poly-gon-shaped tiles in a non-rectangular canvas and enable users to drag a tile in any direction, resize a tile and rotate a tile. In the modification process, most areas between tiles do not overlap, and edges between tiles can be aligned.

(3) We performed user studies and found that our toolkit is convenient and efficient for users to create a customizable hierarchy tile menu for VR/AR environments.

2 Related work

2.1 VR/AR menus

In VR/AR environments, interaction operations such as selection, manipulation, travel, wayfinding, and system control are crucial. Among these, menus serve as the primary means of system control. While there have been numerous studies on selection, manipulation, travel, and wayfinding in VR/AR, research on menus has been relatively limited. Raimund et al. [13] conducted a comprehensive survey and classification of 3D menus based on various criteria.

In VR/AR, graphical menus are more commonly used compared to voice or gesture-based menus. Classic 2D graphical menus have been adapted to the 3D environment of VR/AR, taking into account their familiarity and ease of use for users with minimal training required to confidently navigate them [16].

2.1.1 Tile menu

Tile menus are widely used for their efficiency in selection operation and their feature of supporting users to flexibly translate and scale each tile and arrange their layout at will [5,30,6]. Microsoft has used the style of rectangle tile since Windows 8. Although tile menus are deprecated in Windows 11, there are still tools (such as Live Tiles Anywhere) for creating tile menus. There is also some other work on optimizing packing tiles in a window, which can be seen as a combinatorial optimization problem [19,23].

Tile menus are also engaged in mainstream VR/AR systems, such as Microsoft HoloLens 2. However, they only provide plugins with limited functionality for developers to rearrange tiles in a grid format, which is not enough for developers to create customizable menus. The menu boundary and tile shape in the existing tile menus are also rectangular, and the scalability is poor, which is similar to other typical VR/AR menus. When

submenu items of a menu item are displayed, this menu item will not be visible, and users cannot perceive the global information.

2.1.2 Hierarchical menus

The number of displayed items in virtual graphics menus is crucial for efficient menu navigation and selection operations in VR/AR applications [26]. Hierarchical menus are more suitable for these applications as they reduce the screen space required to display the hierarchy [13]. Existing menu hierarchies include TUISTER [10], virtual windtunnel menus [9], collapsible cylindrical trees [13,12], and cone trees [31], which provide an arbitrary number of sub-items. However, current tile menus in VR/AR lack hierarchical design and display all menu items at once [16]. Treemaps, commonly used for visualizing hierarchical data, can be employed to automatically generate hierarchical menus [32,33]. Although previous works have focused on visualizing hierarchical data using treemaps [39,4,37,29,21,34,38], they do not address modifying the treemap layout. Existing menu hierarchies lack a nested structure like a tree diagram, leading to issues such as occlusion and space occupation in VR/AR displays.

2.2 Graphical layout optimization

Tile menus, like GUIs windows and widgets, are primarily rectangular and designed for rectangular screens. Optimizing tile packing is a combinatorial optimization problem. Kacem et al. [23] proposed an algorithm for dynamically generating 2D rectangular tile menus. It considers the context of use, packs rectangular objects into variable-size screens, and supports menu hierarchies. Dayama et al. [14] explored grid generation for GUIs from four perspectives: layout creation, constraint solving, combinatorial optimization, and interactive generation. They introduced a method based on mixed integer linear programming for diverse grid-based layouts. However, these approaches focus on packing rectangles within windows.

To pack rectangular UI elements in freeform display spaces, Riemann et al. [28] used environmental constraints. Niyazov et al. [26] decomposed interfaces into deformable units and controlled their positions and behaviors, allowing packing of non-rectangular UI elements. To our knowledge, no previous approach addresses readjusting non-rectangular UI elements in freeform display spaces.

2.3 Toolkit for menu or hierarchical view

Several HCI and information visualization toolkits have been proposed and evaluated, including Infovis [17] and Prefuse [20]. Many of these toolkits are relevant to menus in virtual environments. TULIP [8] introduced a menu system using Pinch Gloves, while the gMenu system [16] utilized a stretchable grid as a container for customizable tile menus. Considerations such as layout, style, and personalized design are crucial for interactive menus in VR/AR. Other systems, like the tile-based mixed reality authoring interface [7], PalmType [36], and the menu with circular arrangement [18], offer unique approaches to interface design in VR/AR environments. However, there is a need for a more efficient, flexible, and customizable hierarchical tile menu generation toolkit that supports various shapes in non-rectangular display spaces. Additionally, users should have the ability to interactively adjust tile properties and design layouts according to their preferences or needs. Current toolkits have limitations in supporting personalized design options for users.

3 System description

Given the three above problems: the tile menu can only automatically arrange rectangular tiles within rectangular boundaries; users can only rearrange and resize tiles within a grid format; there is a lack of tools to automatically prototype VR/AR hierarchical tile menus. The goals of our proposed toolkit are:

(1) Automatically pack tiles of different shapes in any shape canvas, e.g., hexagon.

(2) Enable users to readjust tiles interactively (e.g., position, size, shape) and flexibly design and modify various layouts by combining their preferences or needs.

(3) Automatically generate hierarchical tile menu prototypes for VR/AR applications.

The overall framework of our toolkit is shown in Figure 2. The system includes the Generator for automatically generating and modifying menu layouts and the Creator for automatically prototyping menus. Users can use the Generator to generate and modify the menu layout. The Generator uploads the layout file to the server cloud, then the Creator downloads it from the cloud and renders the menu for display. Sometimes programmers take on the role of UI designers. However, UI designers rarely act as programmers because they are usually unfamiliar with the Unity editor, so we deploy the Generator on the web platform and integrate the Creator into Unity as a plug-in.

As shown in Figure 3(a), the user uses the Generator to design a hierarchical menu interface layout on

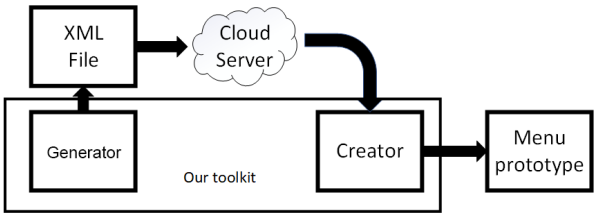


Fig. 2: System framework.

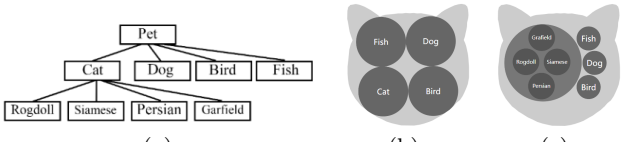
the web page. Then, as shown Figure 3(b), the Creator automatically generates a hierarchical menu prototype in the Unity editor.

(a) (b)

Fig. 3: (a)Generator, (b)Creator.

3.1 Generator

We introduce the Generator with a specific example. As mentioned earlier, our first goal is to automatically generate hierarchical menu layouts with non-rectangular tiles within certain boundaries. Figure 4 shows an example of a hierarchical tile menu. Figure 4(a) shows a simple pet hierarchy, and Figure 4(b) and Figure 4(c) are two layouts of a hierarchical tile menu with a cat head border shape and a circular button shape. According to the previous method, in addition to designing the child node layout of the Pet node (Figure 4(b)), it is also necessary to design the child node layout of the other non-leaf nodes when they are focused. For example, Figure 4(c) shows the layout of one of the non-leaf nodes – the node "Cat".



(a) (b) (c)

Fig. 4: (a) tree structure of menu, (b) layout of node "Pet" and (c) layout of node "Cat".

In order to simplify repetitive work, the Generator can automatically generate the layouts of each non-leaf node according to the tree structure, boundary shape, and tile shape selected by the user. Then users can preview and switch the layout of each node in the Generator, interactively re-adjust the tiles (e.g., location, size, shape) in a specific layout, and flexibly modify the layout by combining tiles with their preferences or needs.

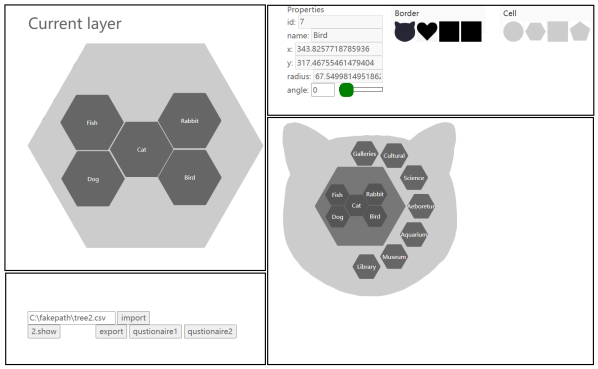


Fig. 5: Generator interface.

The interface of the Generator includes four parts, as shown in Figure 5 the upper left shows the local view of the menu, which is used to switch the displayed non-leaf node and render the layout of the current non-leaf node. It allows users to adjust the layout of the current node. The lower left is the menu bar for input and output. The properties of the currently displayed node (including id, name, location, radius, and angle) can be modified on the upper right, and the global boundary shape and cell shape can be switched. The lower right shows the global view of the entire menu, the same as the content displayed in a VR/AR device.

3.2 Creator

The second goal is to prototype VR/AR hierarchical tile menu based on the designed menu layout automatically. In this paper, we take HoloLens 2 as an example and implement the Creator as a plug-in for HoloLens 2 in Unity3D.

The Creator can be divided into two parts: Scripts and Resources. Scripts are the core part of the system, including reading XML files and storing the layout of each non-leaf node, binding the clicking event of each non-leaf node, and rendering the layout of a specific node. Resources are a variety of related resources, such as background images, tile icons, and prefabs (a built-in resource type of Unity). The Creator can automatically generate a menu prototype based on the XML file, which can be previewed in Unity3D or interacted with gestures in HoloLens 2.

4 Generating menu layout

As mentioned above, there are two requirements when generating interface layout: in order to generate a hierarchical menu layout conveniently and quickly, we need to generate non-rectangular tiles within non-rectangular boundaries automatically; in order to support users to adjust and design the menu layout according to their needs, we need to set constraints among tiles within the non-rectangular boundary, so that the tiles can be edge-aligned for easy combination.

In the following, we introduce how to automatically pack tiles of different shapes in a canvas of any shape to generate a hierarchical tile menu layout, and then introduce a method that supports users to adjust tiles and flexibly create various layouts interactively.

4.1 Automatically generation method of hierarchical menu layout

In this section, we start with the input format, then explain how to automatically calculate the hierarchical menu layout, output format, and implementation.

4.1.1 Input

We take tree structure data, bounding container, and button shape as input since hierarchical menus mainly consist of these essential elements. The input format of the tree structure is a series of menu items specified by the user. Herein, each menu item contains the node's unique identifier, the name, the weight, and the unique identifier of its parent node. The bounding container is represented as an array of polygon vertices, each element storing the two-dimensional coordinates of the vertices. The button shape can be selected as a circle or regular polygon.

4.1.2 Method

The Generator generates a treemap layout using a modified circular treemap algorithm [40]. The algorithm arranges sub-circles within a circle space to create a unified layout for hierarchical data. To improve computation efficiency, we make the following enhancements:

We modify the treemap generation method to calculate the positions and radii of all cells simultaneously, starting from a normalized initial circle. The correct positions are rendered linearly during display.

For the convenience of description, we define the layout of a node as the attributes (position, size, rotation angle) of its child cells within a circular boundary. The actual circle represents the boundary when rendered, and the actual layout represents the cell attributes within the rendered circle. The normal packing layout represents the layout of a node, and the focused packing layout represents the layout when the node is focused.

Using the disk packing algorithm [40], we compute the initial layouts of the "pet" and "cat" nodes simultaneously (Figure 7(a) and Figure 7(b)).

The Creator renders the actual layout of the "cat" node within its parent node "Pet" (Figure 7(c)) by applying the offset and scaling based on the initial circle and actual circle properties.

4.1.3 Implementation

First, for each non-leaf node T, we calculated the cell partition of its child nodes, according to the tree structure of T. Since the shapes of cells in circular treemap are circular, therefore, we specified a circle $C = (p, r)$ as the cell boundary for T, where p is the center of C and r is the radius of C. Then, we performed the following processing for each non-leaf node synchronously by Algorithm 1.

Algorithm 1: Packing Layout Algorithm.

Input: non-leaf node T.
Output: the normal layout of T.

1. Get the number n of child nodes, let each child node as a site, and get the weight of each child node as weight of corresponding site. Scatter n sites inside the boundary circle C randomly.
2. Divide boundary circle C by variational disk packing algorithm in [35]. Thus, we get circle $C_i = (p_i, r_i)$, i=1,. . . ,n, which is corresponding to each site.
3. Calculate and save the transformation relationship of position, size and orientation of each C_i relative to C.

Next, we took the data structure shown in Figure 4(a) as an example, combined with Figure 8, to introduced the process of generating the entire hierarchical tile menu under the premise of a given boundary container (cat head) and button shape (regular hexagon).

In the initialization phase, for all non-leaf node T, we computed the normal packing layout P_T by Algorithm 1 in parallel, where each node corresponds to the initial circle C. Figure 7(b) shows the normal packing layout of the non-leaf node "cat". For each child node T_i of T, we temporarily enlarged the weight of T_i and calculated the focused packing layout P_{T_i} of its parent node T_j, such as the focused packing layout of node "cat", see Figure 7(a). We saved the packing layout as a transformation relation of position, size, and orientation of each C_i relative to C, where C_i represents each

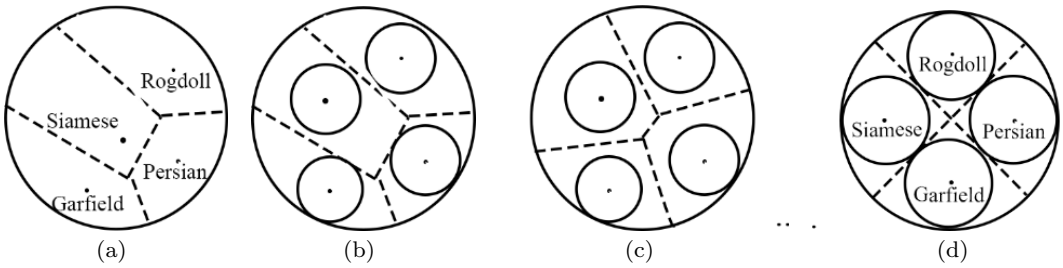


Fig. 6: The interface generation process of a non-leaf node. (a) the initial power diagram. (b) the state after re-computing the location of each site. (c) recalculated power diagram. (d) final circle packing result.

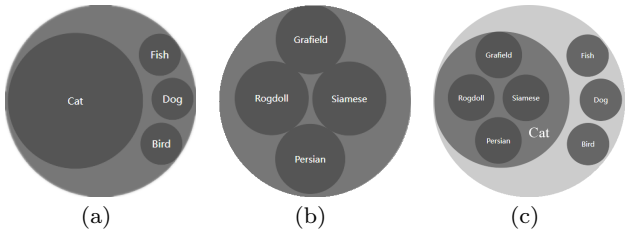


Fig. 7: (a) and (b) are the initial layout of the "pet" and "cat" and (c)the actual normal layout of node "Cat" and the actual focused layout of node "Cat"

child node of T. The process of the initialization phase is shown in Algorithm 2.

In the interactive phase, for the initial display interface and a given boundary container (cat's head shape), the root node T_0 corresponds to this shape, and we calculated the inscribed circle C_{T_0} of the shape (Figure 8(a)). Then we mapped the normal packing layout P_{T_0} to C_{T_0} according to the parameter information between the unit circle C and C_{T_0}, such as the radius ratio. Then we computed the inscribed polygon (Figure 8(c)) of each circle (Figure 8(b)) in P_{T_0}, and rendered the initial interface (Figure 8(d)).

For the current interface, assuming node T_i is focusing and the parent of T_i is T_j, we mapped the focused packing layout P'_{T_i} to C_{T_j} . Then we computed the inscribed polygon (Figure 8(f)) of each circle (Figure 8(e)) in P_{T_j}, and got the interface (Figure 8(g)). Herein, we calculated the inscribed circle C_{T_i} of the shape of node T_i (Figure 8(h)), then we mapped the normal packing layout P_{T_i} to C_{T_i} . Then we computed the inscribed polygon (Figure 8(j)) of each circle (Figure 8(i)) in P_{T_i}, and rendered the interface after node T_i was clicked (Figure 8(k)).

4.2 Readjust hierarchical menu layout

We explore a new approach to keep a stable and predictable layout during adjust-layout user interaction, i.e., dragging a tile within its parent tile with drag-move-drop mouse interaction and scaling a tile with a slide-mouse wheel interaction.

4.2.1 Interface layout modification

After the interface is automatically generated, designers may also need to make interactive optimization modifications, including general and structural modifications.
General modification. Users can perform modification operations (translation, rotation, and scaling of cells) for each cell in each node, see Figure 9. We define three operation interactions: the user drags the center of the cell with one finger to move the cell (see Figure 9(a)); the user taps and holds the edge of the cell with one finger to move towards/away from the center of the cell, and changes the cell size (see Figure 9(b)); the user taps the edge of the cell with one finger and rotates around its center of the cell to rotate it (see Figure 9(c)). This node modification does not need the recalculation of the treemap but only needs to be saved to the modified node structure and passed to its descendants. Hence, the whole modification process is in real-time.
Structural modification. The Generator enables users to modify the hierarchy structure on the generated interface (see 10).

Tiles can be added, deleted, or moved with tap and drag interaction. The circle packing algorithm only needs to be called at most twice per modification.

4.2.2 Support algorithm

The circumcircle ensures that sibling cells of a particular shape do not overlap when they are generated, see

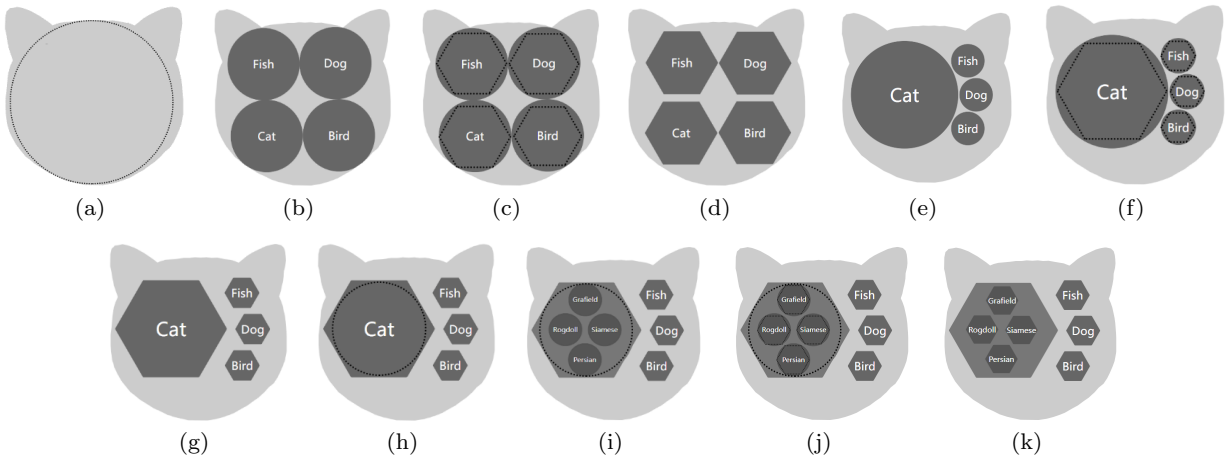


Fig. 8: Generation process of a hierarchical interface with cat's head boundary shape and regular hexagon button shape.

Algorithm 2: Calculating layout packing for initialization phase.

Input: the tree structure data, boundary container, and button shape.
Output: the normal packing layouts and the focused packing layouts.

```
foreach T in all non-left nodes do
    Compute and output the normal packing layout P_T by Algorithm 1 in parallel.
    foreach child node T_i of T do
        Increase the weight of T_i to half of the sum weight of all the child nodes of T.
        Calculate and output the focused packing layout P'_{T_j} of T by Algorithm 1.
    end
end
```

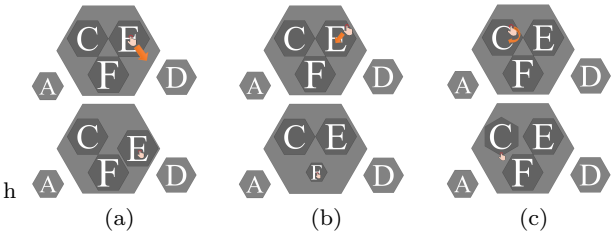


Fig. 9: Three cell modification methods (the above is before modification, and the following is after modification. (a), (b) and (c) are translation, zoom and rotation, respectively).

Fig. 10: moving node C from the child node of root to node B. (a) before modification, (b) after modification.

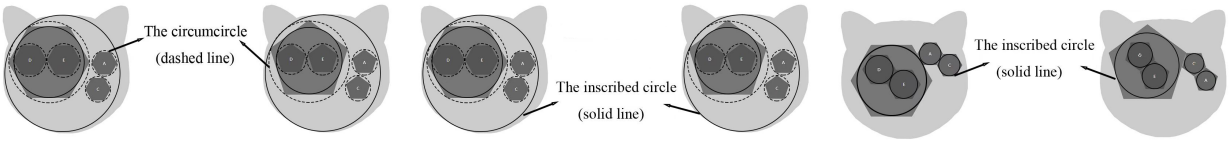
Figure 11(a) and Figure 11(c). The inscribed circle ensures that sibling cells of a specific shape are all within their parent nodes when generated, see Figure 11(b). The inscribed circle ensures that sibling cells of a specific shape do not overlap as much as possible during adjustment, see Figure 11(c).

We use the modified Circlepacker algorithm to support the layout adjustment. Circlepacker models [25] assume graphical objects (typical nodes of a network) are part of a physical space ((x, y) position) and get mechanical, gravity, and electrostatic forces applied to them.

We assume each child of the parent cell is assigned a circle with two position parameters $c_i(x, y)$ and a radius r_i. Typically, we render the static layout of this set of cells generated by the circular treemap. When a user modifies the state of a tile, the Generator passes the position and radius of the inscribed circle of the tile to the Circlepacker model [25], obtains the state of other tiles in real-time, and renders them on the canvas. When the user stops the operation, all tiles stop moving, and the layout remains stable.

4.3 Output

As mentioned, the layout of the hierarchical menu includes the normal packing layout of each non-leaf node



(a) (b) (c)

Fig. 11: (a) the circumscribed circle constraint of the cell during automatic generation (b) the inscribed circle constraint of the cell during automatic generation (c) the inscribed circle constraint of the cell during modification.

and the focused packing layout of each non-leaf node when it is focused. Therefore, the XML file output by our toolkit includes the boundary shape polygon, button shape, and the normal layout of each non-leaf node and the focused layout of each non-leaf node.

5 Creating menu prototype

In order to automatically create menu prototypes in VR or AR applications, we need to implement a hierarchical menu prototype creator that does not require user programming. The Creator is implemented as a plug-in in Unity. The Creator reads the XML file output by the Generator, creates a tree structure based on the data in the file, saves the normal packing layout and focused packing layout of each node, and binds click events for each non-leaf node.

The initial menu shows the normal packing layout of the root node, see Figure 8(d).

Figure 12(a) shows the initial interface. When the user clicks the "Cat" button, the Creator first renders the actual focused packing layout of the parent node "Root" of "Cat" (see Figure 12(b)) and then render the actual normal packing layout of the "Cat" node (see Figure 12(c)).

6 User study

We performed a user study to evaluate the perceived usability and user experience using our developed toolkit to create a hierarchical menu prototype for the AR system.

6.1 Participants

In this study, we take people with programming experience or design experience as participants to get their professional opinion and feedback on our toolkit. A total of 35 participants (10 women and 25 men) voluntarily participated in this study and were assigned to one of two levels of expertise in Unity programming (expert versus novice). Participants were students and professionals aged 14-42 (M=25.16, SD=6.94) with computer science, mechatronics and information engineering, digital media, and art backgrounds.

6.2 Apparatus and Setup

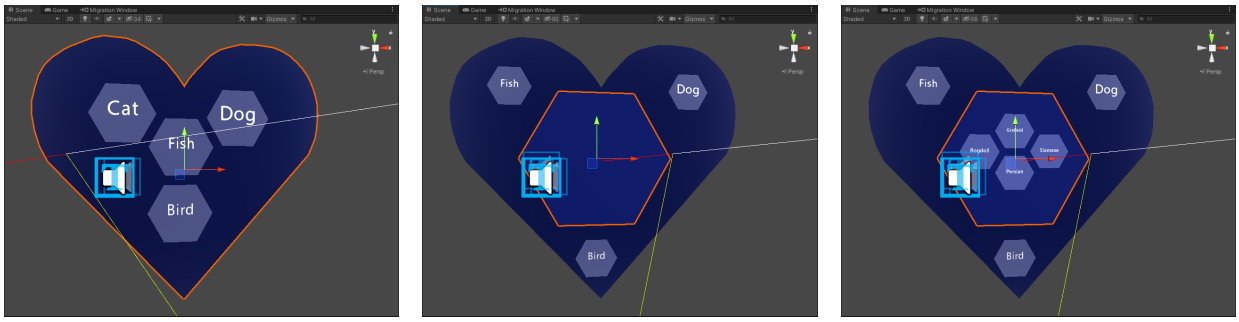
Our prototype AR system used the second version of Microsoft's HoloLens (development edition) and a Dell Alienware Area-51m laptop (1920×1080 17'3 screen) running Windows 11 with a wired HHKB HYBRID TYPE-S keyboard and a wired Logitech G502 mouse. We connected a LECOO M2411 display (1920×1080 screen 23'8 screen) to the laptop via SaiKang HDMI to DVI adapter cable, and we connected the laptop to a local TP-Link AD1550 router via Ethernet, to which the HoloLens connected via WiFi.

6.3 Study design and procedure

Before participants started the task, they completed a questionnaire containing demographic information and questions about their knowledge and experience related to designing, programming, and HoloLens 2. Participants then experienced a hierarchical tile menu designed by professional UI designers using our tool. They had 15 minutes to explore the menu and afterwards completed a PSSUQ questionnaire. A discussion about the hierarchical tile menu followed.

Next, participants were divided into two groups based on their Unity programming experience. Group A consisted of participants with Unity programming experience, who were given 30 minutes to design the required UI using the toolkit. Participants in both groups rated items about heuristics, creativity, system functioning, and usability on a seven-point Likert scale ((Figure 13)). The study concluded with a discussion about the tool and its usage in the two working environments. The entire test lasted about 90 minutes for each participant.

Results of the PSSUQ questionnaire showed that the scores of PSSUQ and its sub-scales were all higher



(a) (b) (c)

Fig. 12: Generation process of a hierarchical interface with cat's head boundary shape and regular hexagon button shape.

than the upper limit of reference score [27] (see Table 1), which suggested hierarchical tile menu had a high usability.

The discussion about the hierarchical tile menu showed that most participants were interested in hierarchical menus. The participants said: "I think the hierarchical information is clear" (P1) and "I like the tree structure, and the hierarchy is clear" (P17).

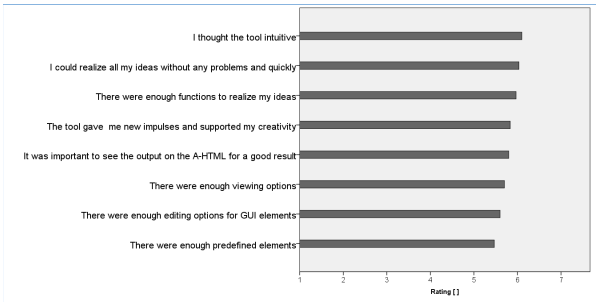


Fig. 13: Means of the rating for the final questionnaire's items (7-point Likert scales; 1=strongly disagree to 7=strongly agree).

The final questionnaire (Figure 13) and the final discussion showed that the participants were overall satisfied with the usability and functionality of the tool. One participant without program experience said: "It is straightforward to (use the toolkit to) complete the task" (P2), and another said: "The operation is effortless and easy to learn" (P3). An experienced programmer from another group validated this. He admitted, "There is no obstacle to operate (this tool)" (P4). In general, the final questionnaire (Figure 13) and the discussion revealed that the significant benefit of the tool was practical since it supported the "easy" (P2, P5, P9, P14, P18, P20, P24, P30, P33, P34), "convenient" (P4, P5, P6, P8, P9, P10, P21, P22, P25, P27) and "comfortable" (P5, P13, P21, P28) operation. One participant outlined this advantage: "I prefer the neatly arranged menu in the tool when it is automatically generated" (P5). Nevertheless, guidance for users is crucial, as one participant pointed out. He said: "the dissatisfaction is that I sometimes get confused about the left and right (mouse) button operations (in the design process)" (P4). It should "give some operation tips" (P2) and "give some guidance" (P22).

Figure 14 presents the means of the ratings for the statements concerning heuristics, creativity, and usability. We aggregated the data by these dimensions. The data showed that programmers (M = 5.579, SD = .205) outperforms non-programmers (M = 5.424, SD = .159) for heuristic. A t-test revealed that the difference is not statistically significant, t = 1.459, p = .175. While the rating of creativity was rather similar for programmers (M = 5.736, SD = .320) and non-programmers (M = 5.733, SD = .320), t = .013, p = .990, the usability was rated higher for non-programmers (M = 6.045, SD = .273) than for programmers (M = 5.934, SD = .105), and the difference was not statistically significant, t = .750, p =.482.

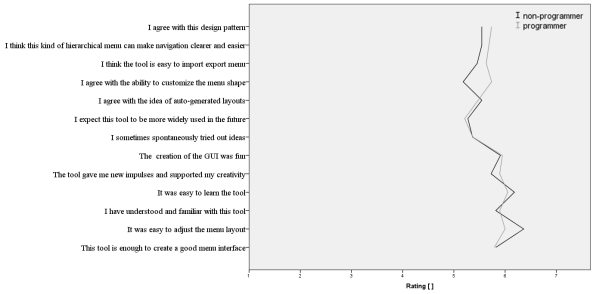


Fig. 14: Mean values of the rating for the programmers and non-programmers. The items are rated on 7-point Likert scales (1=strongly disagree to 7=strongly agree).

TABLE 1 PSSUQ questionnaire results

	Our work	Lower limit	Normal limit	Upper limit
Overall quality	5.52	2.62	2.82	3.02
System quality	6.02	2.57	2.80	3.02
Information quality	5.24	2.79	3.02	3.24
Interface quality	5.10	2.28	2.49	2.71

Participants had different points of view in the discussion due to their different levels of familiarity with Unity programming. According to the usability rating, programmers rated the tool higher for design patterns and functional modules. ("If I need to change the hierarchical structure of the menu frequently, including adding, deleting, etc., I prefer this tool" (P23), "No need to write code yourself, function integration is more effective for specific tasks. It is easy to use for non-professionals, and the threshold is low" (P33)). The most recurring argument for non-programmers is "convenience". When discussing the comparison with their usual design menu, the participants with design experience said: "I think the hierarchical information is clear" (P1) and "I like the tree structure, and the hierarchy is clear" (P17). This finding echoes our observations and rating trends during the study period.

7 Discussion

7.1 Comparison with previous toolkits in related work

The previous toolkit mainly focused on auto-generate menus or interfaces in the virtual environment, and they did not support packing non-rectangular UI elements in non-rec tangular display space [16,8,22,18, 15,36]. There are works on packing rectangles in non-rectangular space, but they are either geometric algorithms [19,23] or interfaces that do not support user's input and realignment of layout [26]. Using the Unity engine to develop hierarchical menus of non-rectangular-shaped buttons in non-rectangular display spaces is complex and time-consuming. To our knowledge, no previous approach simultaneously tackles the challenge of automatically generating hierarchical menus with non-rectangular shaped buttons and supporting users to interactively re-adjust the tiles (e.g., location, size, shape) and flexibly design various layouts by combining tiles with their preferences or needs.

7.2 User experience of using our toolkit

User studies on our customizable hierarchy tile menu toolkit yielded valuable insights. Participants, including programmers and UI designers, expressed interest in hierarchical menus, finding them clearer and more straightforward. Programmers scored higher on heuristics but lower on usability compared to UI designers. However, no significant differences were found in ratings for heuristic, creativity, and usability. The discussions with programmers revealed their better understanding of design nuances but also highlighted the need to consider features available in the Unity editor. Taking both programming and UI design perspectives into account is crucial for enhancing the toolkit's functionality and usability.

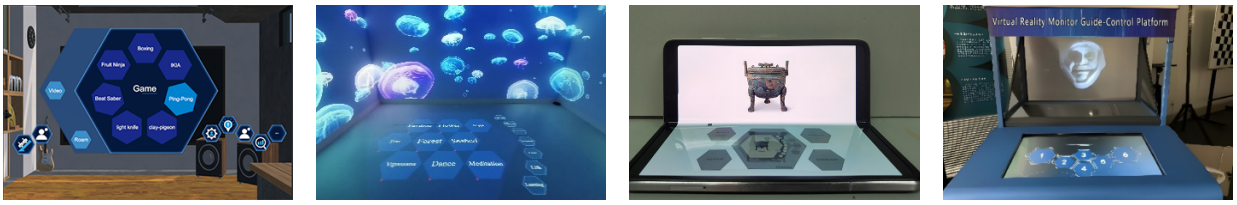
7.3 Running time

This method needs to call the circle packing algorithm n times when generating the interface (n is the number of non-leaf nodes in the tree structure) during initialization. These calculated layouts will be stored for user interaction. The circle packing algorithm - the space partition algorithm adopted by circular treemap - has linear time complexity and only takes 0.005s for 115 sites (CPU speed of 2.41 GHz,512 MB RAM) [7]. The biological taxonomic dataset from the Encyclopedia of Life (EOL) [2] has at most 713 non-leaf nodes, and modified circular treemap costs at most 3.565 seconds to generate the treemap layout of EOL, which is a typical tree structure used for visualization. However, the tree structure of the hierarchical menu we are working on is usually simple. Therefore, in practice, the generation time will be lower than this upper limit.

To check whether the user adjusts the menu in real time, we add nodes one by one within a cardioid boundary and invite users to judge whether there is a sense of delay, which will be clear when the number of nodes is more significant than 83. In practical applications, menu items usually will not exceed this upper limit. According to the 7 ± 2 principles [40], there are generally no more than 9 menu items in the navigation menu. Within this range, our tools can support users to make real-time adjustments.

7.4 Application diversity

Currently, the most comprehensive VR/AR solution (such as HTC Vive and HoloLens 2) is based on controller-



(a) (b) (c) (d)

Fig. 15: Interface menu created for different devices: (a) HMD, (b)CAVE, (c) smart phone, (d) Guide-control platform.

based or hand gesture-based ray-casting interaction [22]. Therefore, our tool's regular polygon-shaped buttons are more suitable for ray-casting interaction in virtual environments. Although we use HoloLens 2 as a representative of VR/AR devices in a user study, in practical applications, our menu can be extended to various VR/AR devices and even some other devices and has been used to develop some other application systems for different fields, see Figure 15. The menu generated in this article serves as a prototype for the programmer to develop a specific system. Developers can choose the tree structure to generate the menu prototype and bind the click event of the leaf button to a specific task.

7.5 Limitation

Since our purpose is not to replace commercial software, we have yet to implement all potential functions, such as modifying text formats, modifying button textures, and other functions. Some participants also suggested "adding the function of modifying the background and button color" and "adding animation effect when clicking". Although these features can be implemented in future work, they are not the focus of this study. Currently, in our prototype, UI designers only participate in our tool's generation and export of hierarchical menus. In contrast, binding click events of menu items in hierarchical menus with scenes or objects in AR programs still require programmers.

8 Conclusions

This paper proposes a toolkit for generating customizable layered tile menus for VR/AR systems. The toolkit utilizes a modified circular treemap algorithm to automatically create hierarchical menu layouts on various-shaped canvases. Users can adjust tile properties in real time and design menu layouts according to their preferences. A prototype implementation of the toolkit demonstrates its convenience and efficiency in creating customizable hierarchy tile menus, enhancing the usability of VR/AR systems. Notably, user programming is not required for generating the hierarchical menu layouts. Future work includes refining the tool's functionality, integrating it into VR/AR system development studios, and incorporating features like using HoloLens 2 to scan physical surfaces as canvases. This research aims to enable common users to develop their Metaverse application systems.

References

1. live-tiles-anywhere. https://apps.microsoft.com/store/detail/live-tiles-anywhere/9NR7QQK712PL (Access from July 3, 2022)
2. Encyclopedia of life. http://www.eol.org (Access from June,13, 2022)
3. Akkersdijk, S.M., Brandon, M., Jochmann-Mannak, H., Hiemstra, D., Huibers, T.: Imagepile: an alternative for vertical results lists of ir-systems (2011)
4. Auber, D., Huet, C., Lambert, A., Renoust, B., Sallaberry, A., Saulnier, A.: Gospermap: Using a gosper curve for laying out hierarchical data. IEEE transactions on visualization and computer graphics **19**(11), 1820–1832 (2013)
5. Belkacem, I., Pecci, I., Martin, B.: Smart glasses: A semantic fisheye view on tiled user interfaces. In: 2016 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 1405–1408. IEEE (2016)
6. Bly, S.A., Rosenberg, J.K.: A comparison of tiled and overlapping windows. pp. 101–106. ACM New York, NY, USA (1986)
7. Bowman, D., Wingrave, C., Campbell, J., Ly, V.: Using pinch gloves (tm) for both natural and abstract interaction techniques in virtual environments (2001)
8. Bowman, D.A., Wingrave, C.A.: Design and evaluation of menu systems for immersive virtual environments. In: Proceedings IEEE Virtual Reality 2001, pp. 149–156. IEEE (2001)
9. Bryson, S.: The virtual windtunnel: A high-performance virtual reality application. In: Proceedings of IEEE Virtual Reality Annual International Symposium, pp. 20–26. IEEE (1993)
10. Butz, A., Groß, M., Krüger, A.: Tuister: a tangible ui for hierarchical structures. In: Proceedings of the 9th international conference on Intelligent user interfaces, pp. 223–225 (2004)

11. Cohen, E.S., Smith, E.T., Iverson, L.A.: Constraint-based tiled windows. IEEE computer graphics and applications **6**(5), 35–45 (1986)
12. Dachselt, R., Ebert, J.: Collapsible cylindrical trees: A fast hierarchical navigation technique. In: Information Visualization, IEEE Symposium on, pp. 79–79. IEEE Computer Society (2001)
13. Dachselt, R., Hübner, A.: Three-dimensional menus: A survey and taxonomy. Computers & Graphics **31**(1), 53–65 (2007). DOI https://doi.org/10.1016/j.cag.2006.09.006
14. Dayama, N.R., Todi, K., Saarelainen, T., Oulasvirta, A.: Grids: Interactive layout design with integer programming. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pp. 1–13 (2020)
15. Deering, M.F.: Holosketch: a virtual reality sketching/animation tool. ACM Transactions on Computer-Human Interaction (TOCHI) **2**(3), 220–238 (1995)
16. Dunk, A., Haffegee, A.: The gmenu user interface for virtual reality systems and environments. pp. 746–753. Springer (2009)
17. Fekete, J.D.: The infovis toolkit. In: IEEE Symposium on Information Visualization, pp. 167–174. IEEE (2004)
18. Gerber, D., Bechmann, D.: The spin menu: A menu system for virtual environments. In: IEEE Virtual Reality 2005, pp. 271–272. IEEE Computer Society (2005)
19. Hart, S.M., Yi-Hsin, L.: The application of integer linear programming to the implementation of a graphical user interface: a new rectangular packing problem. Applied Mathematical Modelling **19**(4), 244–254 (1995). DOI https://doi.org/10.1016/0307-904X(94)00033-3
20. Heer, J., Card, S.K., Landay, J.A.: Prefuse: a toolkit for interactive information visualization. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 421–430 (2005)
21. Horn, M.S., Tobiasz, M., Shen, C.: Visualizing biodiversity with voronoi treemaps. In: 2009 Sixth International Symposium on Voronoi Diagrams, pp. 265–270. IEEE (2009)
22. Hou, S., Thomas, B.H., Lu, X.: Vrmenudesigner: A toolkit for automatically generating and modifying vr menus. In: 2021 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), pp. 154–159. IEEE (2021)
23. Kacem, I., Kadri, I., Martin, B., Pecci, I.: Algorithms on a variable-size rectangular interface. In: 2021 IEEE International Conference on Networking, Sensing and Control (ICNSC), vol. 1, pp. 1–6. IEEE (2021)
24. LaViola Jr, J.J., Kruijff, E., McMahan, R.P., Bowman, D., Poupyrev, I.P.: 3D user interfaces: theory and practice. Addison-Wesley Professional (2017)
25. Mario, G., Georg, F.: circlepacker. https://github.com/snorpey/circlepacker (accessed 13 June 2022)
26. Niyazov, A., Mellado, N., Barthe, L., Serrano, M.: Dynamic decals: Pervasive freeform interfaces using constrained deformable graphical elements. Proceedings of the ACM on Human-Computer Interaction **5**(ISS), 1–27 (2021)
27. Piekarski, W., Thomas, B.H.: Interactive augmented reality techniques for construction at a distance of 3d geometry. In: Proceedings of the workshop on Virtual environments 2003, pp. 19–28 (2003)
28. Riemann, J., Schmitz, M., Hendrich, A., Mühlhäuser, M.: Flowput: Environment-aware interactivity for tangible 3d objects. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies **2**(1), 1–23 (2018)
29. Rios-Berrios, M., Sharma, P., Lee, T.Y., Schwartz, R., Shneiderman, B.: Treecovery: Coordinated dual treemap visualization for exploring the recovery act. Government Information Quarterly **29**(2), 212–222 (2012). DOI https://doi.org/10.1016/j.giq.2011.07.004
30. Robertson, G.G., Mackinlay, J.D.: The document lens. In: Proceedings of the 6th annual ACM symposium on User interface software and technology, pp. 101–108 (1993)
31. Robertson, G.G., Mackinlay, J.D., Card, S.K.: Cone trees: animated 3d visualizations of hierarchical information. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 189–194 (1991)
32. Scheibel, W., Limberger, D., Döllner, J.: Survey of treemap layout algorithms. In: Proceedings of the 13th international symposium on visual information communication and interaction, pp. 1–9 (2020)
33. Scheibel, W., Trapp, M., Limberger, D., Döllner, J.: A taxonomy of treemap visualization techniques. In: VISIGRAPP (3: IVAPP), pp. 273–280 (2020)
34. Scheibel, W., Weyand, C., Döllner, J.: Evocells-a treemap layout algorithm for evolving tree data. In: VISIGRAPP (3: IVAPP), pp. 273–280 (2018)
35. Shneiderman, B., Plaisant, C., Cohen, M.S., Jacobs, S., Elmqvist, N., Diakopoulos, N.: Designing the user interface: strategies for effective human-computer interaction. Pearson (2016)
36. Wang, C.Y., Chu, W.C., Chiu, P.T., Hsiu, M.C., Chiang, Y.H., Chen, M.Y.: Palmtype: Using palms as keyboards for smart glasses. In: Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services, pp. 153–160 (2015)
37. Wattenberg, M.: A note on space-filling visualizations and space-filling curves. In: IEEE Symposium on Information Visualization, 2005. INFOVIS 2005., pp. 181–186. IEEE (2005)
38. Wettel, R., Lanza, M.: Visualizing software systems as cities. In: 2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, pp. 92–99. IEEE (2007)
39. Xiao, S., Bian, Y., Yang, C., Meng, X., Liu, S., Li, M., Sun, Q., Qi, G., Liu, J., Zhou, N., Wei, Y.: Optimal device choice and media display: A novel multimedia exhibition system based on multi-terminal display platform. Procedia Computer Science **129**, 103–109 (2018). DOI https://doi.org/10.1016/j.procs.2018.03.056. 2017 INTERNATIONAL CONFERENCE ON IDENTIFICATION,INFORMATION AND KNOWLEDGEIN THE INTERNET OF THINGS
40. Zhao, H., Lu, L.: Variational circular treemaps for interactive visualization of hierarchical data. In: 2015 IEEE Pacific Visualization Symposium (PacificVis), pp. 81–85. IEEE (2015)