Fractal Adaptive Menus: Concept, Engineering, and Use Case

Alaa Eddine Anis Sahraoui
Jean Vanderdonckt
alaa.sahraoui@uclouvain.be
jean.vanderdonckt@uclouvain.be
Université catholique de Louvain, LouRIM
Louvain-la-Neuve, Belgium

Suzanne Kieffer
suzanne.kieffer@uclouvain.be
Université catholique de Louvain
Institut Langage et Communication (IL&C)
Louvain-la-Neuve, Belgium

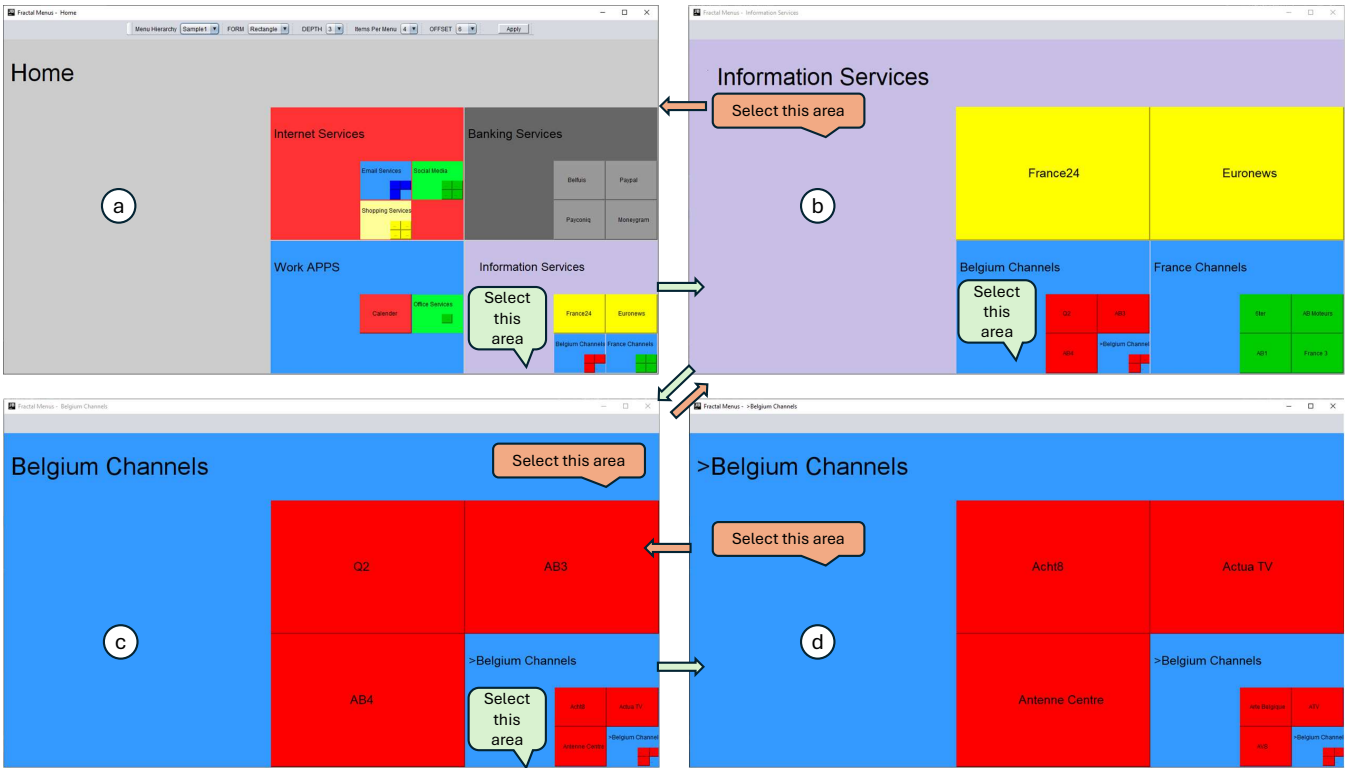

Figure 1: Fractal Adaptive Menu: the main screen opens with the hierarchical decomposition of menus, sub-menus, and items as recursive rectangular areas (a); selecting any area zooms in on this area to reveal the sub-menus, recursively: *e.g.*, selecting the (a) purple area leads to (b), selecting the (b) blue area leads to (c) and (d), iteratively; selecting the main area outside the sub-menu returns to the previous level: *e.g.*, selecting the (d) blue area returns to (c), then to (b), then to (a).

Abstract

The literature is replete with studies that have explored, analyzed, and tested different techniques for ensuring the adaptivity of a menu in graphical user interfaces, mainly by maintaining the format of a menu bar, its pull-down menus, and its sub-menus, while varying its presentation. Instead of restricting adaptivity to this format, adaptivity could be ensured by exploiting the entire display space available in full-screen or intermediate-screen mode. Inspired by fractal geometry, we motivate and define the concept of a fractal adaptive menu, a graphical adaptive menu that benefits from three properties useful for adaptivity: self-similarity, recursive navigation, and hierarchical menu organization. We explain how to engineer this menu for graphical user interfaces and validate this approach with a case study in machine control for a steel mill. In addition to being adaptive to the screen resolution, a fractal adaptive menu is particularly beneficial when menu navigation is repetitive or frequent in a deep hierarchical organization. Proprioceptive memory can make it easier to remember which menu areas to select during a long crossing of the hierarchy.

EICS Companion '25, Trier, Germany

ACM ISBN 979-8-4007-1866-3/2025/06
https://doi.org/10.1145/3731406.3734978

CCS Concepts

• **Human-centered computing** → **Graphical user interfaces**; *Touch screens*; **Ubiquitous and mobile devices**; Tablet computers; • **Software and its engineering** → *Software prototyping*; • **Mathematics of computing** → **Functional analysis**; • **Theory of computation** → **Computational geometry**.

Keywords

Adaptive user interface, Fractal generator, Fractal geometry

ACM Reference Format:
Alaa Eddine Anis Sahraoui, Jean Vanderdonckt, and Suzanne Kieffer. 2025. Fractal Adaptive Menus: Concept, Engineering, and Use Case. In *Companion Proceedings of the 17th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS Companion '25), June 23–27, 2025, Trier, Germany.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3731406.3734978

1 Introduction

The literature is full with works that have studied adaptive menus in a Graphical User Interface (GUI), in the form of introducing original adaptive menus [1, 4, 5, 8, 9, 16, 28], modeling them [11, 19], comparing them [6, 17], and incorporating them in a design space [7, 21]. There are at least 49 different Graphical Adaptive Menus (GAMs) [40], most of which are variations on the classic menu format of a menu bar attached to pull-down menus, themselves associated with sub-menus. Maintaining this format is commendable for preserving overall usability [17, 30], primarily consistency of presentation even when behavior changes [16], maximizing predictability [22], reducing the displacement of items subject to adaptivity [29], minimizing potentially negative effects [25], retaining selection accuracy [22], and enabling generation [39]. For example, menu items subject to adaptivity can be highlighted [4]. Fisheye menus [5] vary item font size according to distance from the focus area, the region in which the cursor is located. In most of these cases, the overall menu format remains the same.

A small proportion of these adaptivity techniques challenge this classic format by imagining new shapes and sizes that expand the field of investigation and design spaces of GAMs [7, 16, 21], which are a common and frequent example of adaptive GUIs. For example, hyperbolic menus break the straitjacket of the classical format by displaying the menu hierarchy in a hyperbola that expands and collapses according to the navigation and the cursor location [40].

Menu adaptivity becomes even more acute in the context of multiple GUIs [20] (which require different adaptations depending on the platform and its screen resolution) and multi-platform interfaces [12] (which multiply the adaptations by the number of screens to be defined, even in responsive design). The effect of the menu size on user satisfaction [2], actual or perceived usability [33], performance [18] and, above all, efficiency is well known [10], especially for mobile devices [33]. However, varying the menu size can go hand in hand with varying the shape, to create a different kind of adaptivity. Instead of restricting oneself to the classic menu format, exploiting the entire available display space represents an opportunity to be explored for both desktop platforms (*e.g.,* for maximized or intermediate windows) and mobile platforms (*e.g.,* smartphones or tablets), not to mention the list format.

In the field of *fractal geometry* [27], the *self-similarity* property of a fractal image refers to its characteristic of appearing similar at different levels of magnification [35]. This means that when zooming in on a fractal, smaller sections of the image resemble the overall structure, often repeating indefinitely [24]. Self-similarity can be exact, where each scaled-down portion is an identical copy of the whole, or approximate, where the similarity is present but not perfectly replicated [32]. This property is evident in mathematical fractals [24, 26], such as the Mandelbrot set [27], snowflakes [31], and coastlines or other geographical forms [3], where patterns recur at progressively smaller scales.

To address the above issues, this paper defines the concept of a fractal adaptive menu, a graphical adaptive menu that benefits from three properties useful for adaptivity inspired by fractal geometry: self-similarity, recursive navigation, and hierarchical organization. We then explain how to engineer this menu for GUIs and validate this approach with a case study in machine control for a steel mill.

2 Related Work

This section first reviews selected work related to GAMs and then delves into fractal geometry, with some applications.

2.1 Graphical Adaptive Menus

GAMs have been extensively studied [40] to maintain or enhance usability [30] by adapting menu structures and presentation based on context of use [5, 9], usage frequency [6, 19], and cognitive load [11]. Various adaptive menu designs have been proposed, each with unique advantages and limitations, costs, and benefits [25]. The fractal menu introduces a novel approach by leveraging principles of fractal geometry to address some of these limitations.

Traditional hierarchical menus organize items into nested structures [28], requiring users to navigate through multiple levels to access specific functions. While effective for categorizing extensive option sets, they often lead to increased interaction time and cognitive load due to deep navigation paths. Users may experience "menu hunting," struggling to recall the exact path to a desired function.

Split menus [36] reorder or highlight frequently used items to improve efficiency *e.g.,* by placing commonly accessed items at the top while maintaining the full menu below. While beneficial for task efficiency, users may find the reordering disruptive or unpredictable [22, 37], leading to frustration when item locations change unexpectedly [29]. *Ephemeral adaptation* [19] employs gradual onset techniques to reduce visual search time while maintaining spatial consistency [16]: predicted items appear to draw attention, while others fade in gradually. This method has been shown to improve menu selection performance by guiding user attention without altering item positions. Closer to our study, a *mega-menu* (Fig. 2-left) is a large, rectangular, expandable menu that displays many items at once, often organized into clearly defined categories, columns, and sometimes enhanced with icons to provide visual cues. This design allows users to quickly scan through a broad range of links and sub-menus without needing to click through multiple layers of navigation. In contrast to this non-adaptive menu, a *square menu* (Fig. 2-right) [1] features a grid-like layout where navigation options are presented as uniformly sized square tiles, offering a more streamlined and minimalist appearance.

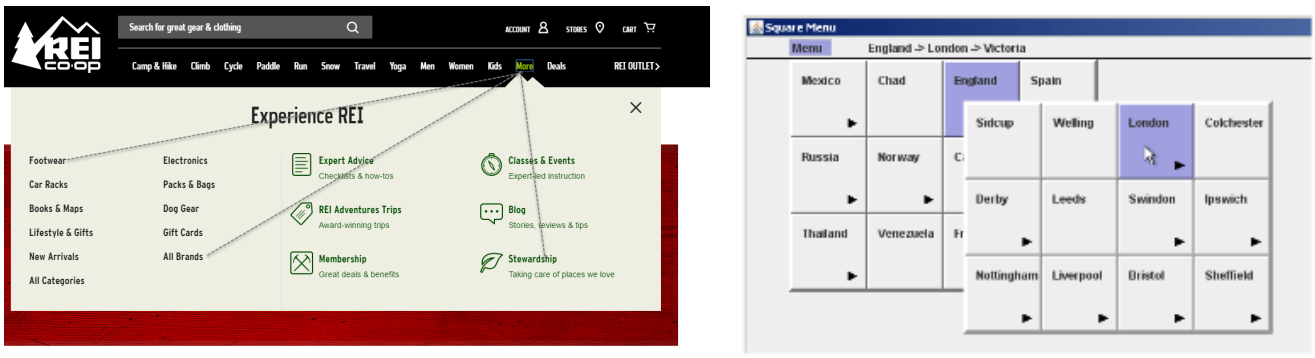

Figure 2: Two graphical menus using rectangular shapes: (*left*) a rectangular mega menu, (*right*) a square menu [1].

2.2 Fractal Geometry in HCI

Fractal geometry has emerged as a valuable tool in human-computer interaction research. It has been applied to recognize brain states from EEG signals, enabling neurofeedback applications for concentration and stress management [38]. Fractal-based algorithms have also been used to create lifelike artificial forms, with studies showing that varying fractal dimensions can influence user perceptions of naturalness and appeal [31]. In computer science, fractal geometry has found applications in image compression and virtual reality environments, allowing for the reproduction of complex natural patterns [34]. Additionally, the concept of self-similar fluctuations has been proposed as a basis for designing cognition-friendly interactions, with the suggestion that friendly interactions should exhibit antipersistent, self-similar fluctuations within a specific range [13]. These diverse applications suggest the potential of fractal geometry to enhance various aspects of GUIs. There is really no genuine consideration of fractal geometry to adaptive GUIs in general and GAMs in particular to test whether the properties enjoyed by a fractal image could be successfully applied to menus.

3 Concept of Fractal Adaptive Menu

By strict mathematical definition [35], a fractal is a geometric shape that has symmetry of scale, meaning that you can zoom in to an arbitrary depth and see the same pattern repeated. A *Fractal Adaptive Menu* or *Fractal Menu* for short, is a GAM type engineered on the three fundamental principles of fractal geometry:

(1) *Self-Similarity*: The menu structure and format repeat at multiple levels, where sub-menus resemble the main menu in presentation and behavior. Imagine a circular or grid-based menu where each main category appears as a node (or a tile). When the user clicks/taps a node, it zooms in to reveal subcategories, which maintain the same visual style but on a smaller scale. The deeper the user navigates, the more detailed the options become. For example, in Fig. 1, each level is represented by a rectangular area containing smaller rectangles for sub-menus.
(2) *Recursive Navigation*: Thanks to animated transitions [14], users can zoom in to reveal more detailed items or options, and zoom out to see a broader view. For example, in Fig. 1, the user can activate any visible sub-menu by selecting the corresponding rectangular area (either by clicking on it with a mouse or by touching it on a tactile surface) and can come back to the previous level by selecting the area outside the sub-menus. In this way, users located at a level n access any sub-menu at subsequent levels $n-1$, $n-2$, etc. if they exist and if they are displayed and return to the upper level $n+1$ if it exists. Fig. 1 shows examples where forward navigation in the hierarchy is supported by "Select this area" in the sub-menu structure and backward navigation is supported by "Select this area" outside the sub-menu structure. This mechanism is referred to as *incremental retreat* [15] when navigation is reversed level by level.
(3) *Hierarchical Organization*: Items are structured in a tree-like form, allowing users to navigate naturally through different levels of menu items. For example, in Fig. 1, the first level (a) displays four sub-menus, *i.e.,* red, grey, blue, and purple; each menu is further decomposed into sub-menus if relevant, *e.g.,* the purple menu "Information services" is activated (b) to display two menu items (*i.e.,* "France24" and "Euronews") and two sub-menus (*i.e.,* blue with red and blue with green). Selecting the "blue and red" sub-menu in (b) further activates the sub-hierarchy of this menu (c), and so forth.

In addition to satisfying these three principles, a fractal adaptive menu is expected to offer two features important for supporting adaptivity to various GUI screen resolutions:

(1) *Adaptive Complexity*: The menu dynamically adjusts its level of detail based on available screen real estate and user interaction, ensuring efficient access to frequently used functions. For example, in Fig. 1, top right, the green sub-menu has only four leaf nodes, thus creating an adaptive area for four, while the red sub-menu is further decomposed into three levels, not all of them being displayed to preserve browsability (when menu items are visible on-demand) instead of observability [23] (when menu items are directly visible).
(2) *Space Efficiency*: Unlike traditional nested menus that occupy large amounts of screen space, a fractal adaptive menu condenses information while keeping it accessible. For example in Fig. 1, all rectangular areas are computed recursively based on a fractal generator (*i.e.,* the pattern image subject to self-similarity) [34].

4 Engineering Fractal Adaptive Menus

Since our fractal adaptive menu should result in an adaptive multi-platform menu that can adapt to varying screen resolutions and where selection depends on the user's preference and usage, we created such a menu based on rectangular areas. To maintain its fractal dimension low, thereby reducing the menu complexity, we limit the number of items/levels per menu as well as its view depth in the 4-6 range. Opportunistic, manual programming of a fractal adaptive menu would not be flexible enough to accommodate various properties. To avoid this approach, we modelled our fractal generator as a Lindenmayer system, or L-System [3], a rewriting system exploiting a formal grammar, composed of symbols that can be used to make strings, a set of production rules that expand each symbol into some larger string of symbols, an initial "axiom" string from which to begin construction, and a mechanism for translating the generated strings into the fractal image.

For this purpose, we rely on the Turtle representation for L-Systems [26], which comprises symbols such as: F = Move forward (draw a line forward along the turtle's heading), G = Move forward (without drawing a line), + = Turn left of a certain angle α. Fig. 3-left shows the first steps of a rule, which consists of drawing forward, turn right 90°, drawing forward, turn right 90°, and drawing forward.

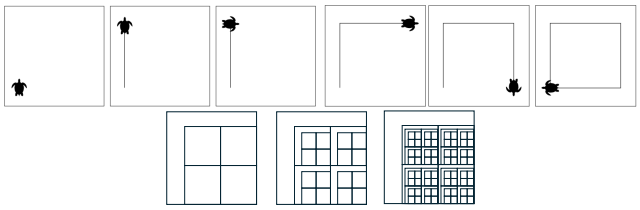

Figure 3: Two examples of a Turtle interpretation: the F—F—F—F rule with α=90 (*top*) and the three first iterations for our menu (*bottom*).

Iteration 1 of our rectangular fractal menu has the following Turtle interpretation (Fig. 3-bottom for the three first iterations):

- Alphabet: FG
- Axiom: GGGG-GGGG-GGGG-GGGF; Note that other axioms could be defined, such as GGG-GGG-GGG-GGF, FFF-FFF-FFF-FFF, FFF-FFF-FFF-FF-FF-FF-F-FF-F-F-FF-FF
- Rule: F = G++GG+GG+GG+G+GG+G+G+GG
- Constants: none
- Turning angle: 90°;
- Starting angle: 90°

This gives it a fractal dimension close to 1. For subsequent iterations, each small rectangle will be replaced by a reduced version of the global shape. To engineer the whole process of producing a fractal adaptive menu (Fig. 4-left), we developed a Java applet[1] in Java Platform 8 that: (1) parses the menu hierarchy stored as an XML file into an object, (2) uses the menu item object along with the L-system describing the fractal to be interpreted by the fractal generator, and (3) renders the menu in a resizable window. The parser distinguishes two types of menu items: *final* and *non-final* menu items, representing a leaf menu item and a sub-menu item, respectively. To complete the fractal menu, we used recursion and the L-system. The distinction between final and non-final items helps us create an exit condition for the recursion. The menu is initialized by the fractal generator using the MenuItem from the parser and a size value based on the current screen resolution or window dimensions if not full screen. The following pseudo-code describes this sequence:

[1]An alternative way to implement a fractal adaptive menu in a web interface is to use React.js for UI components and D3.js for zooming and hierarchical layout.

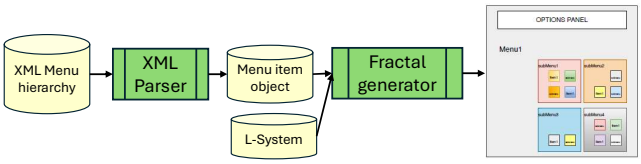

Figure 4: Flow diagram for engineering a rectangular fractal adaptive menu.

Algorithm 1 : Generate a Rectangular Fractal Panel

```
Class MenuS
    MenuS(mi, s):                          ▷ mi = MenuItem from parser,
        panelColor ⟵ mi.getColor();
        panelText ⟵ mi.getText();
        setPanelDimension(s);
        nextSize ⟵ s/2;
        if mi.isFinal() == false then
            for MenuItem m in mi.getMenuItems() do
                addSubPanel(MenuS(m,nextSize));
            end
        else
            addSubPanel(MenuS(m,nextSize));
        end
        assembleSubPanels();
        setColorAndBorderLayout();
        addListeners();
```

Algorithm 2 : MenuItemPanel listener

```
Function listener(event):
    MenuItem chosen ⟵ event.getSource().getMenuItem();
    MenuItem parent ⟵ chosen.getParent();
    if event.isLeftMouseButton then
        FractalGenerator.execute(chosen);
    else
        FractalGenerator.execute(parent);
End Function
```

Item selection is achieved by capturing mouse events: left-mouse click to zoom into the selected item area to display its children and right-mouse click to zoom out of the selected item area to display its parent menu. On any click event, we get the MenuItem of the clicked item if it was a right-mouse click and/or its parent MenuItem if it was a left mouse click. The MenuItem is loaded for display via the FractalGenerator. Depending on the platform, other item selection techniques could be considered and implemented, such as touch selection, eye tracking, or EEG-based navigation [38]. Deeper

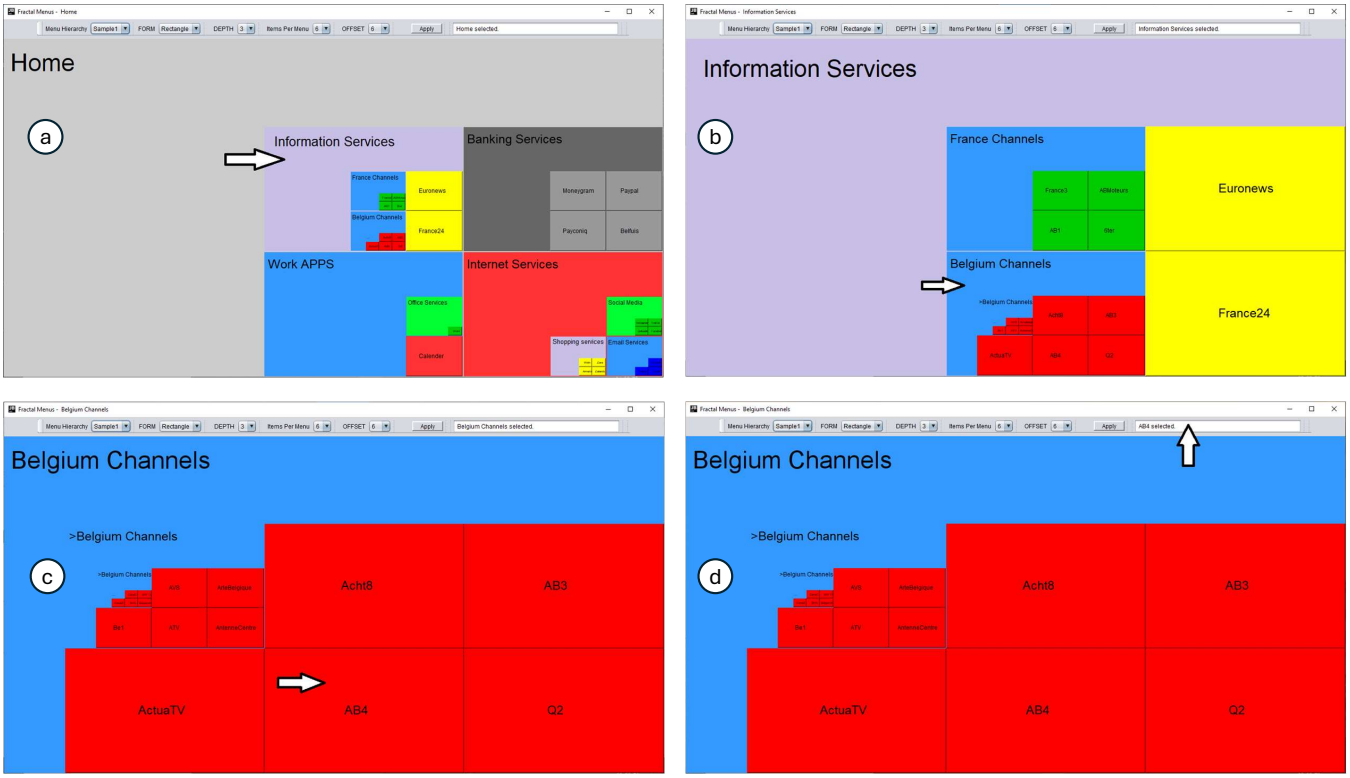

Figure 5: Fractal Adaptive Menu with parameter control: opening the "Information Services" sub-menu (a), then the "Belgian channels" sub-menu (b), then selecting "AB4" channel (c). Selecting any parent level in the breadcrumb (d). Note the recursive sub-menus in (c) and three levels displayed in (a).

items or sub-menus can be accessed by sequentially selecting the sub-menus leading to them. Menus or sub-menus with more than 4 items will have three of the items shown, and the rest will be on a subsequent screen accessible via the fourth menu item area to establish browsability [23]. In sum, the core features of this engineering are: (1) a zoomable GUI where users can click on any rectangular area to zoom into sub-menus, similar to how zooming works in fractal viewers [34], (2) a seamless transition between levels supported by animated transitions between levels, (3) a dynamic content loading where sub-menus are dynamically generated only when they are needed depending on available screen resolution; and (4) a breadcrumb to help users track their location in the hierarchical organization, located in the top right position in Fig. 5.

Fig. 5 reproduces an interactive session where the fractal adaptive menu is used with its control bar displayed at the top of the window for flexible control: Menu hierarchy enables the end user to switch from one menu to another, Form selects the shape used for the fractal generator, Depth specifies the number of menu levels that can be displayed on demand, Items per menu gives the threshold for the maximum number of items displayed at once for each level, and Offset specifies the space between the main area and the subarea containing the sub-menus. The top right position contains the breadcrumb of the menu for direct selection. In this way, the fractal adaptive menu provides end users with more control to customize its behaviour based on user's needs, preferences, and additional constraints posed by the computing platform.

For example, in Fig. 5, the fractal adaptive menu is operating with Depth=3 (up to three levels of sub-menus displayed at once) and Items per menu=6 (up to six items per menu in each rectangular area at one level). Fig. 5-a shows the menu resulting from applying these parameters, in which the end user selects the "Information Services" area, which is then magnified in Fig. 5-top right. The end user then selects the "Belgium channels" area that is in turn zoomed in (Fig. 5-c). Since a maximum number of 6 items can be displayed at once, the top right area in this figure is itself displayed to satisfy this constraint, recursively. If the user selects this area, it will then be magnified as well. If the user selects "AB4", a final item, it will be selected. Fig. 5-d shows a case where the end user returns to any other parent level by selecting the item in the breadcrumb. Note that Fig. 5-(c) shows a screenshot where the sub-menu ">Belgium channels" is recursively displayed with two deeper levels thanks to available screen space. Other areas contain final items that do not require any further display. Similarly, Fig. 5 displays up to three levels of the hierarchical organization of the menu, when such sub-menus exist. Final items are simply displayed in their respective areas, whose surfaces are equivalent at that level. Of course, sub-menus are displayed in smaller areas, thus making them harder to select, but keeping them observable.

5 Real-World Use Case

Possible applications of a fractal adaptive menu include: GUIs of interactive software, such as advanced design tools, development environments, or content creation software; gaming Menus with game settings or inventories where users can expand or collapse categories seamlessly; information visualization by exploring hierarchical datasets or network graphs, and touchscreen interfaces When traditional drop-downs and lists may be impractical.

We hereby describe a real-world project where a fractal adaptive menu has been prototyped, developed, and tested. The use case takes place in a company specializing in the protection of steel through zinc coating, a process essential to industries such as automotive manufacturing. This continuous production process involves high temperatures, heavy machinery, and a demanding physical environment. The company operates 24/7 with rotating teams of workers, including team leaders, quality technicians, and line operators. Each team is responsible for ensuring that the production line runs smoothly and meets stringent safety standards.

A key procedure within this workflow is the preparation of the line before producing parts that require high surface quality. This preparation involves completing a detailed checklist, with thirty interdependent checks carried out by different workers stationed at specific points along the line. The checklist is used to coordinate actions, verify equipment status, and document responsibility. It plays a central role in quality assurance and team communication.

As part of a broader effort to digitize critical procedures and reduce reliance on paper forms, the checklist was selected for digital implementation. A digital interface was therefore developed to replicate and improve upon the existing form. The interface takes the form of a fractal menu, a hierarchical and recursive navigation structure particularly suited to complex, multi-actor procedures. Each item of the original checklist is represented as a distinct node within the menu, preserving the formal organization of the task while enabling dynamic interaction and real-time data entry.

The fractal menu is aligned with three key structural and interactional properties required by the task.

(1) First, *self-similarity* ensures consistency across levels: each menu and sub-menu follows the same visual and functional logic, supporting quick orientation and reducing learning time. Whether accessing high-level categories or fine-grained checklist items, users encounter the same interaction.
(2) Second, the menu supports *recursive navigation* through animated transitions. Users can zoom in to focus on a specific set of checks or zoom out to gain an overview, facilitating flexible access to information while maintaining context. The transitions match the pace and attention span of workers operating under physical and cognitive constraints.
(3) Third, the menu naturally supports a *hierarchical organization of items*. The checklist itself is structured around a tree of responsibilities and spatial locations on the production line. This tree maps directly onto the menu structure, where each node corresponds to a functional area or actor, and sub-nodes correspond to the associated verification tasks. In this way, the fractal menu offers not only a digital replica of the original form but also a more efficient and cognitively coherent interface for navigating distributed tasks.

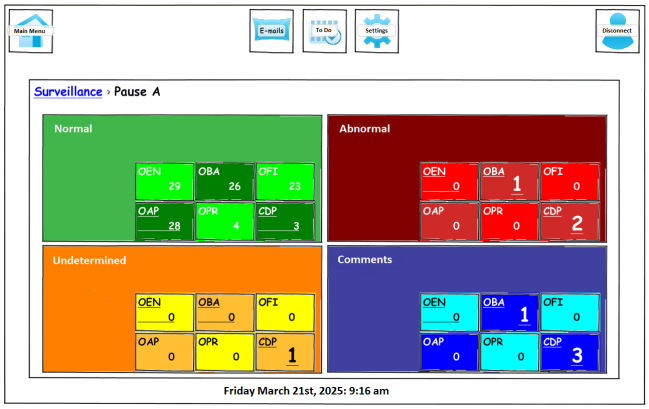

Figure 6: A Fractal Adaptive Menu.

Fig. 6 reproduces a screenshot of an interactive wireframe of the fractal adaptive menu obtained in this real-world use case to ensure machine surveillance, which classifies the end users' tasks into four categories: normal conditions (depicted in green in (Fig. 6) which are the most frequent cases; abnormal conditions (depicted in red), undetermined conditions (depicted in orange), and freeform comments (depicted in blue). Each area is decomposed into subcategories, displayed as sub-menus with the number of occurrences for each sub-task. For example, the "CDP" sub-task contains three items to handle (the abbreviations are standard domain codes).

6 Conclusion and Future Work

This paper introduces the original concept of a fractal adaptive menu, a graphical adaptive menu that satisfies three properties useful for adaptivity inspired by fractal geometry: self-similarity, recursive navigation, and hierarchical organization. Among these properties, self-similarity is the key property since the shape used by the fractal generator is automatically applied at any level of the menu organization on display. Adaptivity is ensured by the shape generator that automatically produces the layout of the menu (both main area and sub-areas) depending on constraints imposed by the screen resolution (in full-screen mode) or the window dimensions (in intermediary-screen mode). We outlined a process for engineering such a fractal adaptive menu by parsing an XML file specifying its hierarchical organization and feeding a formal definition of the fractal as a L-system to render the menu at run-time. To keep the end user in control of this display, a bar enables the end user to control the value of all parameters used to display the menu, mainly the depth level and the number of menu items per level, which gives more flexibility for customization. We validate this approach with a use case study from a real-world project. We believe that a graphical adaptive menu is particularly appropriate in domains where the navigation and item selection are repetitive and frequent in a deep hierarchical organization. This remains to be demonstrated empirically, which is our next step. Finally, we release the Java code of the applet developed for this purpose.

Open Science. We share a public repository with the software developed for this paper, along with the use case. Since our fractal menu is publicly available, it can be reused to perform other use cases for reproducibility or incremental research.

Acknowledgments

Alaa Sahraoui and Jean Vanderdonckt are supported by the EU EIC Pathfinder-Awareness Inside challenge Symbiotik project (1 Oct. 2022-30 Sept. 2026) under Grant no. 101071147. Suzanne Kieffer and Jean Vanderdonckt also acknowledge the support of the Virtuoso research project funded by Plan Marshall of DGO6, Région Wallonne, Belgium, under Grant no. 6608 (ref. THIM/6608). The authors would like to thank the EICS '25 reviewers for their constructive feedback on earlier versions of this manuscript.

References

[1] David Ahlström, Andy Cockburn, Carl Gutwin, and Pourang Irani. 2010. Why It's Quick to Be Square: Modelling New and Existing Hierarchical Menu Designs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. ACM, New York, NY, USA, 1371–1380. https://doi.org/10.1145/1753326.1753534

[2] Khalid Al-Omar and Dimitrios Rigas. 2009. The Effect of Size of Personalised Menus on User Satisfaction. In *Proceedings of the 11th WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering* (Athens, Greece) *(MMACTEE'09)*. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 322–327. http://dl.acm.org/citation.cfm?id=1949006.1949062

[3] Shahzoda Anarova, Shakhlo Sadullaeva Azimbayevna, Golib Berdiev, Adhamjon Toxtasinov Ilhomjon Ogli, and Maftuna Ismoilova Ilxom Kizi. 2024. Geometric Modelling of Fractal Structured Objects based on Integration of R-Function And L-System Methods. In *Proceedings of the Cognitive Models and Artificial Intelligence Conference* (undefinedstanbul, Turkiye) *(AICCONF '24)*. Association for Computing Machinery, New York, NY, USA, 172–181. https://doi.org/10.1145/3660853.3660898

[4] Liat Antwarg, Talia Lavie, Lior Rokach, Bracha Shapira, and Joachim Meyer. 2013. Highlighting items as means of adaptive assistance. *Behaviour & Information Technology* 32, 8 (2013), 761–777. https://doi.org/10.1080/0144929X.2011.650710 arXiv:https://doi.org/10.1080/0144929X.2011.650710

[5] Benjamin B. Bederson. 2000. Fisheye menus. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology* (San Diego, California, USA) *(UIST '00)*. Association for Computing Machinery, New York, NY, USA, 217–225. https://doi.org/10.1145/354401.354782

[6] Sara Bouzit, Gaëlle Calvary, Denis Chêne, and Jean Vanderdonckt. 2016. A comparison of shortcut and step-by-step adaptive menus for smartphones. In *Proceedings of the 30th International BCS Human Computer Interaction Conference: Fusion!* (Poole, United Kingdom) *(HCI '16)*. BCS Learning & Development Ltd., Swindon, GBR, Article 26, 12 pages. https://doi.org/10.14236/ewic/HCI2016.26

[7] Sara Bouzit, Gaëlle Calvary, Denis Chêne, and Jean Vanderdonckt. 2016. A Design Space for Engineering Graphical Adaptive Menus. In *Proceedings of the 8th ACM Symposium on Engineering Interactive Computing Systems* (Brussels, Belgium) *(EICS '16)*. ACM, New York, NY, USA, 239–244. https://doi.org/10.1145/2933242.2935874

[8] Sara Bouzit, Gaëlle Calvary, Denis Chêne, and Jean Vanderdonckt. 2017. Polymodal Menus: A Model-based Approach for Designing Multimodal Adaptive Menus for Small Screens. *Proc. ACM Hum.-Comput. Interact.* 1, EICS, Article 15 (June 2017), 19 pages. https://doi.org/10.1145/3099585

[9] Sara Bouzit, Gaëlle Calvary, Denis Chêne, and Jean Vanderdonckt. 2019. Interface adaptivity by widget promotion/demotion. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (Valencia, Spain) *(EICS '19)*. Association for Computing Machinery, New York, NY, USA, Article 18, 6 pages. https://doi.org/10.1145/3319499.3328237

[10] Sara Bouzit, Gaëlle Calvary, Joëlle Coutaz, Denis Chêne, Éric Petit, and Jean Vanderdonckt. 2017. The PDA-LPA design space for user interface adaptation. In *Proceedings of the 11th International Conference on Research Challenges in Information Science* (Brighton, United Kingdom) *(RCIS 2017)*. IEEE, Piscataway, NJ, USA, 353–364. https://doi.org/10.1109/RCIS.2017.7956559

[11] Andy Cockburn, Carl Gutwin, and Saul Greenberg. 2007. A Predictive Model of Menu Performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '07)*. ACM, New York, NY, USA, 627–636. https://doi.org/10.1145/1240624.1240723

[12] Benoît Collignon, Jean Vanderdonckt, and Gaëlle Calvary. 2008. Model-Driven Engineering of Multi-target Plastic User Interfaces. In *Proceedings of Fourth International IEEE Conference on Autonomic and Autonomous Systems* (Gosier, Guadeloupe) *(ICAS 2008)*. IEEE Computer Society, Piscataway, NJ, USA, 7–14. https://doi.org/10.1109/ICAS.2008.37

[13] Balaram Das. 2001. Cognition friendly interaction: A concept of partnership in human computer interaction. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 11, 3 (09 2001), 632–640. https://doi.org/10.1063/1.1383548 arXiv:https://pubs.aip.org/aip/cha/article-pdf/11/3/632/18302300/632_1_online.pdf

[14] Charles-Eric Dessart, Vivian Genaro Motti, and Jean Vanderdonckt. 2012. Animated Transitions Between User Interface Views. In *Proceedings of the International Working Conference on Advanced Visual Interfaces* (Capri Island, Italy) *(AVI '12)*. ACM, New York, NY, USA, 341–348. https://doi.org/10.1145/2254556.2254623

[15] Graeme E. Field and Mark D. Apperley. 1990. Context and selective retreat in hierarchical menu structures. *Behaviour & Information Technology* 9, 2 (1990), 133–146. https://doi.org/10.1080/01449299008924230

[16] Leah Findlater and Krzysztof Z. Gajos. 2009. Design Space and Evaluation Challenges of Adaptive Graphical User Interfaces. *AI Magazine* 30, 4 (2009), 68–73. https://doi.org/10.1609/aimag.v30i4.2268

[17] Leah Findlater and Joanna McGrenere. 2004. A Comparison of Static, Adaptive, and Adaptable Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. ACM, New York, NY, USA, 89–96. https://doi.org/10.1145/985692.985704

[18] Leah Findlater and Joanna McGrenere. 2008. Impact of Screen Size on Performance, Awareness, and User Satisfaction with Adaptive Graphical User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. ACM, New York, NY, USA, 1247–1256. https://doi.org/10.1145/1357054.1357249

[19] Leah Findlater, Karyn Moffatt, Joanna McGrenere, and Jessica Dawson. 2009. Ephemeral Adaptation: The Use of Gradual Onset to Improve Menu Selection Performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Boston, MA, USA) *(CHI '09)*. ACM, New York, NY, USA, 1655–1664. https://doi.org/10.1145/1518701.1518956

[20] Elizabeth Furtado, João José Vasco Furtado, Wilker Bezerra Silva, Daniel William Tavares Rodrigues, Leandro da Silva Taddeo, Quentin Limbourg, and Jean Vanderdonckt. 2001. An Ontology-Based Method for Designing Multiple User Interfaces. In *Proc. of 1st Int. Workshop on Multiple User Interfaces* (Lille, France) *(MUI 2001)*. Université des Sciences et Technologies Lille 1. https://www.researchgate.net/publication/2567741_An_Ontology-Based_Method_for_Universal_Design_of_User_Interfaces

[21] Krzysztof Z. Gajos, Mary Czerwinski, Desney S. Tan, and Daniel S. Weld. 2006. Exploring the Design Space for Adaptive Graphical User Interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces* (Venezia, Italy) *(AVI '06)*. ACM, New York, NY, USA, 201–208. https://doi.org/10.1145/1133265.1133306

[22] Krzysztof Z. Gajos, Katherine Everitt, Desney S. Tan, Mary Czerwinski, and Daniel S. Weld. 2008. Predictability and Accuracy in Adaptive User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. ACM, New York, NY, USA, 1271–1274. https://doi.org/10.1145/1357054.1357252

[23] Christian Gram and Gilbert Cockton. 1996. *Design Principles for Interactive Software*. Springer, New York, USA. https://doi.org/10.1007/978-0-387-34912-1

[24] Hideki Koike. 1995. Fractal views: a fractal-based method for controlling information display. *ACM Trans. Inf. Syst.* 13, 3 (July 1995), 305–323. https://doi.org/10.1145/203052.203065

[25] Talia Lavie and Joachim Meyer. 2010. Benefits and costs of adaptive user interfaces. *International Journal of Human-Computer Studies* 68, 8 (2010), 508 – 524. https://doi.org/10.1016/j.ijhcs.2010.01.004

[26] Ivan B. Liss and Thomas C. McMillan. 1987. Fractals with turtle graphics: a CS2 programming exercise for introducing recursion. *SIGCSE Bull.* 19, 1 (Feb. 1987), 141–147. https://doi.org/10.1145/31726.31749

[27] Benoît Mandelbrot. 1982. *The Fractal Geometry of Nature*. WH Freeman, New York, NY, USA. https://www.goodreads.com/book/show/558059.The_Fractal_Geometry_of_Nature

[28] J. Mitchell and B. Shneiderman. 1989. Dynamic Versus Static Menus: An Exploratory Comparison. *SIGCHI Bull.* 20, 4 (April 1989), 33–37. https://doi.org/10.1145/67243.67247

[29] Lauren Norrie and Roderick Murray-Smith. 2016. Investigating UI Displacements in an Adaptive Mobile Homescreen. *Int. J. Mob. Hum. Comput. Interact.* 8, 3 (July 2016), 1–17. https://doi.org/10.4018/IJMHCI.2016070101.oa

[30] Jungchul Park, Sung H. Han, Yong S. Park, and Youngseok Cho. 2007. Usability of Adaptable and Adaptive Menus. In *Usability and Internationalization. HCI and Culture*, Nuray Aykin (Ed.). Springer, Berlin, Heidelberg, 405–411. https://doi.org/10.1007/978-3-540-73287-7_49

[31] Dale Patterson and Daniel Della-Bosca. 2016. Fractal Dimension - A Spatial and Visual Design Technique for the Creation of Lifelike Artificial Forms. In *Artificial Life and Computational Intelligence*, Tapabrata Ray, Ruhul Sarker, and Xiaodong Li (Eds.). Springer International Publishing, Cham, 3–12.

[32] Bogdan Popa, Dan Selișteanu, and Alexandra Elisabeta Lorincz. 2022. Possibilities of Use for Fractal Techniques as Parameters of Graphic Analysis. *Fractal and Fractional* 6, 11 (2022), 686–. https://doi.org/10.3390/fractalfract6110686

[33] Dimitrios Raptis, Nikolaos Tselios, Jesper Kjeldskov, and Mikael B. Skov. 2013. Does Size Matter?: Investigating the Impact of Mobile Phone Screen Size on Users' Perceived Usability, Effectiveness and Efficiency. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services* (Munich, Germany) *(MobileHCI '13)*. ACM, New York, NY, USA, 127–136.

https://doi.org/10.1145/2493190.2493204
[34] Nicoletta Sala. 2009. *Fractal Geometry and Computer Science.* IGI GLobal, Hershey, New York, USA, 385–404. https://doi.org/10.4018/978-1-60566-094-3.ch028
[35] Dietmar Saupe. 2003. *Fractals.* John Wiley and Sons Ltd., GBR, 725–732.
[36] Andrew Sears and Ben Shneiderman. 1994. Split Menus: Effectively Using Selection Frequency to Organize Menus. *ACM Trans. Comput.-Hum. Interact.* 1, 1 (March 1994), 27–51. https://doi.org/10.1145/174630.174632
[37] Choonsung Shin, Jin-Hyuk Hong, and Anind K. Dey. 2012. Understanding and Prediction of Mobile Application Usage for Smart Phones. In *Proceedings of the ACM Conference on Ubiquitous Computing* (Pittsburgh, Pennsylvania) *(UbiComp '12)*. Association for Computing Machinery, New York, NY, USA, 173–182. https://doi.org/10.1145/2370216.2370243
[38] Olga Sourina, Qiang Wang, Yisi Liu, and Minh Khoa Nguyen. 2013. Fractal-Based Brain State Recognition from EEG in Human Computer Interaction. In *Biomedical Engineering Systems and Technologies*, Ana Fred, Joaquim Filipe, and Hugo Gamboa (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 258–272.
[39] Jean Vanderdonckt. 1999. Computer-Aided Design of Menu Bar and Pull-Down Menus for Business Oriented Applications. In *Proceedings of the Eurographics Workshop on Design, Specification and Verification of Interactive Systems* (Braga, Portugal) *(DSV-IS '99)*, David J. Duke and Angel R. Puerta (Eds.). Springer, Berlin, 84–99. https://doi.org/10.1007/978-3-7091-6815-8_7
[40] Jean Vanderdonckt, Sara Bouzit, Gaëlle Calvary, and Denis Chêne. 2020. Exploring a Design Space of Graphical Adaptive Menus: Normal vs. Small Screens. *ACM Trans. Interact. Intell. Syst.* 10, 1 (2020), 2:1–2:40. https://doi.org/10.1145/3237190