Design and Research of Embedded Graphic Devices in Intangible Cultural Heritage Pattern Exhibitions

Li Ding[1], Hongfei Hu[2*]

dingli@eurasia.edu[1], huhongfei@eurasia.edu[2*]

Eurasia University, No. 8, Dong Yi Road, Xi'an, China

Abstract. This paper explores the feasibility and impacts of applying embedded graphic devices to the exhibition of intangible cultural heritage patterns. The study takes embedded graphic devices as a starting point and elaborates on key technologies such as system design, graphic display techniques, pattern digitization, and device adaptation. Simultaneously, from multiple dimensions including user experience, cost-effectiveness, and sustainability, it delves into the potential of this innovative exhibition approach, infusing new vitality into the protection and inheritance of intangible cultural heritage.

Keywords: Intangible Cultural Heritage, Pattern Exhibition, Embedded Graphic Devices, Exhibition Innovation

1 Introduction

Exhibiting intangible cultural heritage patterns involves showcasing traditional designs on fabrics, ceramics, ornaments, and other visual elements that represent the skills, crafts, and arts of specific cultural groups. These patterns convey the culture's history, values, significance, and techniques. Traditional pattern exhibitions face limitations with physical displays, failing to capture intricate details, and printed displays lacking interactivity[1]. Digital devices like projectors and smart TVs are costly, bulky, and not portable, restricting their use in heritage exhibitions[2]. In contrast, embedded graphic devices offer new possibilities. They provide digital displays, interactive interfaces, and immersive experiences, making it easier to understand and preserve intangible cultural heritage. This innovative approach enhances engagement. This study explores using embedded devices in exhibitions, focusing on heritage preservation, particularly in the context of China's intangible cultural heritage like embroidery. These intricate patterns, featuring flowers, birds, landscapes, and figures, demand craftsmanship. Embedded devices can showcase these patterns, enabling zooming, rotation, and even user-generated designs. To achieve this, the device needs vector drawing capabilities, enhancing audience interaction[3]. This paper primarily addresses these technological requirements in system design.

2 KEY TECHNOLOGY AND SYSTEM DESIGN

Based on the previous analysis, addressing the introduction's issues presents the following pressing challenges.

ADDT 2023, September 15-17, Xian, People's Republic of China

DOI 10.4108/eai.15-9-2023.2340882

Diverse and Complex Patterns: Presenting intangible cultural heritage patterns on embedded devices poses challenges due to their diversity and complexity. Meeting the hardware graphics requirements is demanding.

Cost-Efficient Embedded Displays: Urgently required are cost-effective, energy-efficient, and compact embedded graphic display devices. While affordable options exist in the market, they often rely on outdated development approaches like low-level languages, resulting in extended development times and limited graphics support.

Fluctuating Hardware Prices: Embedded hardware platforms' prices fluctuate rapidly. Consequently, adapting software to different platforms with efficiency and affordability becomes challenging.

To tackle these issues, we must innovate continually in technical research and development. This innovation will enhance embedded graphic devices' performance and stability when presenting intangible cultural heritage patterns, contributing to the preservation and promotion of cultural heritage[4].

2.1 JavaScript Runtime for Low Cost Embedded Platforms

JavaScript was chosen for its developer-friendly nature, well-defined standards, and robust testing support. We prioritize its use in project development, particularly for complex intangible cultural heritage displays. Scripting languages like JavaScript simplify intricate project development by reducing the need to manage details. An efficient runtime is crucial, hence our focus on JavaScript[5].

This paper addresses two key technical challenges. Firstly, we emphasize the JavaScript runtime to enhance intangible cultural heritage pattern displays. Secondly, we tackle the JavaScript engine to ensure smooth execution on low-cost, low-power embedded hardware platforms, prioritizing resource efficiency.

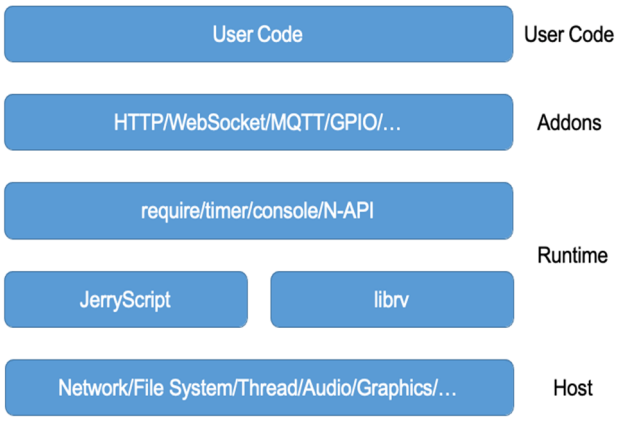

Fig. 1. JavaScript Runtime Architecture

Figure 1 illustrates our JavaScript runtime's layered architecture: user code, addon, runtime (API and JavaScript engine), and HOST (BSP and Hardware). This structure ensures effective execution, with user code as the entry point, addon layer for added functionalities, runtime

layer for JavaScript code interpretation, and HOST layer forming the foundation. This architecture optimizes communication and execution.

The JavaScript engine including its compiler and linker. In accordance with the ECMAScript specifications, the JavaScript runtime implements only generic features that all scripts can use. An application defines the specific features that its own scripts can use through C callbacks. An application that uses the JavaScript runtime is a host in ECMAScript terminology. The JavaScript engine is implemented in C for performance or direct access to native APIs[6]. Thus, it only consumes an embedded platform about 45 KB of free memory, 1 MB of Flash ROM, and runs at 80 MHz.

2.2 Graphics Library Supporting 2D Vector Graphics

Embedded devices should possess vector drawing capabilities in order to effectively showcase the patterns and their formation processes in intangible cultural heritage. Therefore, the support of a vector graphics user interface (GUI) library becomes crucial. Considering the price and performance limitations of embedded devices, supporting 2D vector graphics is currently more feasible. Hence, the main research focus of this paper is to explore 2D vector GUI libraries[7] and mainly describe how to design Canvas path and image drawing.

Canvas paths programmatically construct shapes. They consist of subpaths, each made up of connected points via straight or curved lines. A subpath can be closed, connecting its last and first points with a straight line. Subpaths with just one point are ignored during rendering. A "need new subpath" flag exists in paths. Activating it prompts certain APIs to create a new subpath instead of extending the previous one. When initializing a CanvasPath interface object, ensure the path has zero subpaths. To describe an arc resembling a circle's circumference, points are added to a subpath. This arc starts at a defined angle and ends at another, linked to the previous point with a straight line. A new closed subpath is created to represent the rounded rectangle, using a "radii" parameter, which can be a list or a singular radius, defining the rectangle's corners in pixel units. If it's a list, it follows CSS 'border-radius' property behavior, while a single radius is equivalent to a list with one element[8].

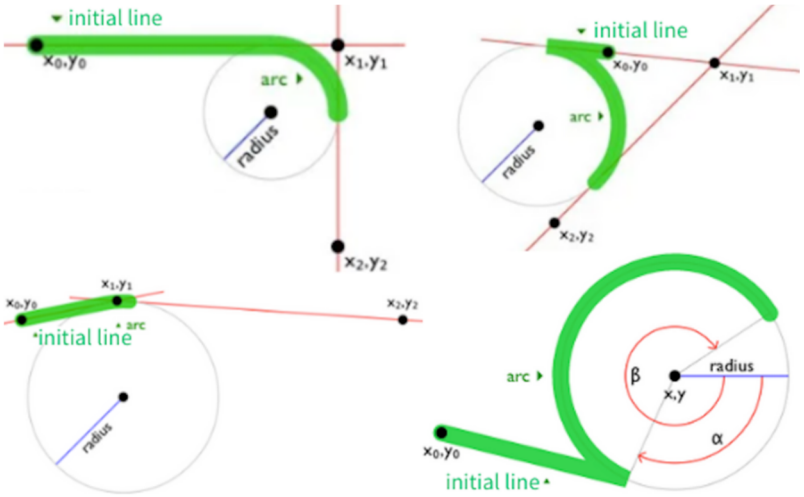

Fig 2. Comnplex Shapes Generation by CanvasPath

In Figure 2, the path's orientation, either clockwise or counterclockwise.

When 'r' in the "radii" parameter is a numeric value, it represents circular arcs with a radius of 'r'. If 'r' within "radii" is defined as an object with {x, y} properties, the corresponding corners become elliptical arcs, with radii 'x' and 'y' equal to 'r.x' and 'r.y', respectively. In cases where the sum of the radii of two corners on the same edge exceeds the edge's length, all radii within the rounded rectangle are scaled by 'length / (r1 + r2)' to maintain proportions. If multiple edges exhibit this property, the smallest scaling factor, aligning with CSS behavior principles, is applied.

Drawing Images, whether GIFs, JPEGs, or PNGs, can be singular files or a collection of JPEGs and PNGs within a folder. The subsequent sections delve into microcontroller graphics basics. They encompass insights on incorporating graphic elements into projects and an overview of various drawing techniques[9].

As illustrated in Figure 3, typically, the dw and dh parameters should default to the values of sw and sh. This interpretation ensures that each CSS pixel in the image corresponds to one unit in the output bitmap's coordinate system. When sx, sy, sw, and sh arguments are omitted, they default to 0, 0, the image's natural width in pixels, and the image's natural height in pixels, respectively. If the image lacks natural dimensions, the concrete object size takes precedence. It's determined using the "Concrete Object Size Resolution" algorithm in CSS, with the specified size lacking fixed width or height constraints. The object's natural attributes align with those of the image argument, and the default object size equals the output bitmap's dimensions.

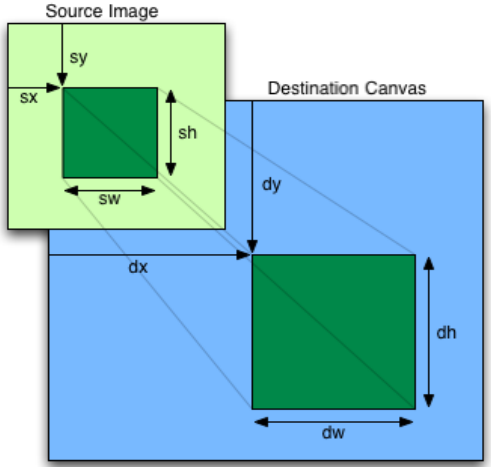

Fig 3. Image Drawing Alghorithm and Methond

When the source rectangle is outside the source image, the source rectangle must be clipped to the source image and the destination rectangle must be clipped in the same proportion.

3 RESULTS AND DISCUSSION

The implementation of embedded graphic devices for the exhibition of intangible cultural heritage patterns presents promising outcomes and opens up new avenues for preserving and showcasing cultural heritage. In this section, we discuss the key results and their implications.

3.1 Graphics Library Supporting 2D Vector Graphics

The inclusion of a graphics library supporting 2D vector graphics has been instrumental in achieving high-quality pattern display as shown in Figure 4. This library allows for the creation of detailed paths and shapes, enhancing the visual representation of intangible cultural heritage patterns. Features such as rounded rectangles and image drawing further enrich the exhibition experience. The ability to scale and clip source images ensures that patterns are displayed accurately, even when dealing with diverse source materials.

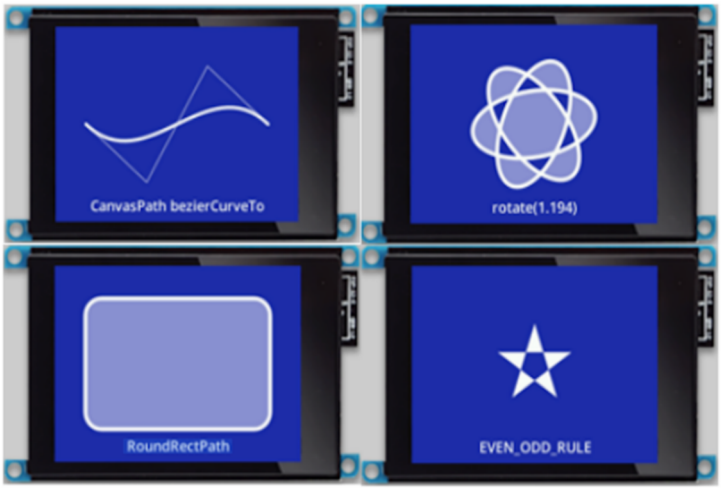

Fig 4. Several Different Paths by 2D Vector Graphic Library

3.2 Enhanced Pattern Display

One of the primary objectives of this research was to provide a platform that can effectively display intricate details of intangible cultural heritage patterns.

Fig 5. Complex Pattern Display of Embroidery

Through the use of embedded graphic devices, we have successfully achieved this goal. Figure 5 shows that these devices allow for zooming and rotating of patterns, enabling audiences to appreciate the nuances and craftsmanship of each design. Moreover, the vector drawing capabilities of the embedded devices provide an interactive experience, allowing users to witness the formation process of embroidery patterns.

4 Conclusions

In conclusion, the application of embedded graphic devices represents a promising innovation in the exhibition of intangible cultural heritage patterns. It addresses the limitations of traditional display methods, offering enhanced pattern visibility, cost-effectiveness, and portability. The combination of JavaScript runtime and a graphics library supporting 2D vector graphics has proven effective in achieving the project's goals. Moving forward, continued research and development in this area can contribute significantly to the preservation and promotion of cultural heritage worldwide.

While the results of this study demonstrate the feasibility and benefits of embedded graphic devices in the exhibition of intangible cultural heritage patterns, there are still avenues for further exploration, such as user experience enhancement, integration with cultural context and collaboration with knowledge sharing.

Acknowledgments. This paper is supported by 2023 Shaanxi Province Social Science Foundation. Project number: 2023J241.

References

[1] Ya Z, Xin X. A Research Review on the Digitization of Intangible Cultural Heritage. Library and Information Service, 61(2): pp. 6-15, January 2017. DOI: 10.13266/j.issn.0252-3116.2017.02.011.

[2] Erturk N. Preservation of digitized intangible cultural heritage in museum storage. Milli Folklor, 16(128), pp. 100-110, December 2020. https://dergipark.org.tr/en/pub/millifolklor/issue/58685/703813.

[3] F. Windhager et al., Visualization of Cultural Heritage Collection Data: State of the Art and Future Challenges, IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 6, pp. 2311-2330, 1 June 2019, doi: 10.1109/TVCG.2018.2830759.

[4] L. D. Gardner, (2018)JavaScript on Things, Manning Publications,Paris, pp. 204-300, https://www.manning.com/books/javascript-on-things.

[5] D. Sin, D. Shin, (2016)Performance and resource analysis on the javascript runtime for iot devices. In: Computational Science and Its Applications. In: 16th International Conference, Beijing, pp. 602-609. https://link.springer.com/chapter/10.1007/978-3-319-42085-1_50.

[6] E. Gavrin, S. J. Lee, R. Ayrapetyan, (2015)Ultra lightweight JavaScript engine for internet of things. In: Companion Proceedings of the 2015 ACM SIGPLAN International Conference on Systems, Pittsburgh, pp. 19-20. https://doi.org/10.1145/2814189.2816270.

[7] Y. L. Shen, S. W. Seo, Y. Zhang, (2010)A low hardware cost 2D vector graphic rendering algorithm for supersampling antialiasing. In: 2010 Second International Workshop on Education Technology and Computer Science. Wuhan, pp.141-144. doi: 10.1109/ETCS.2010.302.

[8] P. Joshi, M. Bourges, K. Russell, and Zh. Mo. (2012) Graphics programming for the web. In: ACM SIGGRAPH 2012 Courses Association for Computing Machinery, New York, , pp. 1–75. https://doi.org/10.1145/2343483.2343491
[9] M. Casario, P. Elst, C. Brown, N. Wormser, C. Hanquez, HTML5 Drawing APIs. In: HTML5 Solutions: Essential Techniques for HTML5 Developers. Apress. https://doi.org/10.1007/978-1-4302-3387-9_6.