Abstract

Based on the GDI+ class library and dual-buffer technology of .NET framework, the software for LED lamps’ layout is developed. On the basis of determining the scheme of layout platform, the metafile class library model by Composite design-set and the process of calling the LED pixels metafile symbol interface are designed firstly. The design method of realizing the LED pixels metafile visual editing on the drawn LED logical pixel based on GDI+ is analyzed. The dual-buffer technology is used to solve the problem of flicker when fast refreshing. The testing results show that the platform can meet the basic function requirements of engineering application, and realize partial adjust on the process of LED lamps layout without distortion.

© 2011 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of Harbin University of Science and Technology

Keywords: Led Lamp; Layout; GDI+; Dual-buffer

1. Introduction

Urban landscape lighting is an important means to beautify the living environment and show the splendors of the city. As the fourth generation product in lighting industry, LED has the characteristics of long life, high luminous efficiency, light colour purity, stability, security, no radiation, low power consumption and shock resistance, which gradually replaces the traditional light source in the landscape lighting and other lighting area [1,2]. The expansion of the LED lamps scale, improvement of landscape design complexity and the integrated requirement of the complex building surface and the LED lamps, lead to the arbitrary spatial relationship of the LED lamps, which makes the uneven distribution of LED

* Corresponding author: Wen Yuanmei. Tel.: +8620-3932-2253.
E-mail address: wen_yuanmei@163.com.
lamps and LED pixels. When making the layout of LED lamps on computer screen, the non-proportion amplification of the LED pixels spacing causes the deformation effect of the actual layout result. So the platform design for LED lamps layout becomes one of the key problems in the heterogeneous LED decorative lights control system [3].

The software for LED lamps’ layout should realize two basic functions: (1) to describe the position of each LED lamp in the layout plane, the LED pixel physical coordinates must correspond to the logical coordinates on the computer screen. (2) to control all the LED lamps properly and achieve data transmission and display, the control logic mapping between controller and LED lamps must be realized.

The representative software platforms for the LED lamps’ layout are: LED Manager, Easy Player and LED Player, which have largely achieved the functions above. Led Manager achieves the LED pixel layout by filling the data table one pixel by one pixel, in which a pixel needs to fill in four parameters (physical and logical coordinates). When dealing with a large project, the complex operation process causes a great many of work [4]. Easy Player supports directed graphic operation, but does not support the partial adjust of the layout documents [5]. Led Player is not open to consumers, which directly provides the layout file if it is the general layout of video wall, when the project is needed to change, the user can do nothing [6]. Combined GDI+ class library with dual-buffer technology, which is powerful in developing general graphical under the .NET framework of Windows platform, this paper has designed a software platform for LED lamps’ layout, which supports animation editing effect and partial adjust on the process of LED lamps layout.

2. The key technology of graphics platform design for LED lamps’ layout

The graphics platform for LED lamps’ layout is designed to be used in the control of heterogeneous LED decorative lights system. The platform has the following features: mapping position and control logic relationship of LED lamps; achieving graphical operation; supporting auto and manual layout together; realizing partial adjust on the process of LED lamps layout without distortion. The technical architecture of layout platform is shown in Fig. 1.

When the LED master-slave logic controller receives a request from client application side, it sets its own parameters according to the request and transfers the request to the LED pixel drawing logic model, which calls the GDI+ metafile class library through the interface function and sends the resulting data to View based on dual-buffer technology, then feeds back to the client application side through pages. The key technologies involved in the design will be described below.

2.1 Design of GDI+ metafile class library

GDI+ is the upgraded version of GDI. It optimizes the performance of GDI and offers more excellent features [7]. The circle, line and other metafiles needed in the design of the platform can be achieved by applying GDI+ to pack different metafile class. To reduce the developing difficulties, the Composite designed-set is used to construct the class library model using metafile as the object. The structure of the model is shown in Fig. 2.

When building a metafile-objected model, the metafile is the Component, simple metafile (Point, Straight line) is the leaf, the complex one (graph) is the Composite. By abstracting the class as well as using the multi-state characteristics of class, interactions between the metafiles class are achieved. The unified external interface of graphs (point, line, surface) is designed and the graphs are packaged as dynamic link library.
2.2 Design of LED pixel drawing logic model

In order to achieve the function of logic pixels changing in proportion to the physical pixels, LED pixel drawing logic model is needed that: ① drawing the logic pixels and signal line quickly; ② editing the metafiles on the drawn logical pixel to realize the continuous zoom, rotate and shift without distortion.

When drawing on the screen based on GDI+, the visual editing is realized by the coordinate transform in GDI+. First, transform the world coordinate to the page coordinate by “world transform”; then, transform the page coordinate to the device coordinate by page transform. The basic ways of metafile transformation based on GDI+ are:

① Zoom: Scaling the parameter value of X-axis and Y-axis, that is:
\[
\begin{align*}
x' &= S_x \cdot x, \\
y' &= S_y \cdot y
\end{align*}
\]  
(1)

② Rotation: Rotating \( \theta \) in counter-clockwise around the origin, that is:
\[
\begin{align*}
x' &= x \cdot \cos \theta - y \cdot \sin \theta \\
y' &= x \cdot \sin \theta + y \cdot \cos \theta
\end{align*}
\]  
(2)

③ Shift: Increasing the starting values of X-axis or Y-axis, that is:
\[
\begin{align*}
x' &= x + T_x \\
y' &= y + T_y
\end{align*}
\]  
(3)

Zoom and rotation changes the position and shape of the metafile from Equation (1) and (2). Shift changes the centre of metafile without changing the shape from Equation (3). When editing the metafile, if the current centre is the transform centre, the computation will be reduced. Affine transformation is the right choice, in which linear transformation with the centre of metafile not changed is implemented first and then shift transformation is implemented. The length and angle will change in the affine transformation[8].

In order to eliminate the possible distortion existed in the composite affine transformations, the three-dimensional homogeneous coordinates is introduced. If the two-dimensional coordinates of metafile based on GDI+ is \((x, y)\), the three-dimensional homogeneous coordinates is \((hx, hy, h)\), in which \(h\) is a real number. When \(h=1\), under the three-dimensional homogeneous coordinates, the transformation matrixes of zoom, rotation, and shift are expressed as follow:

\[
S = \begin{bmatrix}
S_x & 0 & 0 \\
0 & S_y & 0 \\
0 & 0 & 1
\end{bmatrix}, \\
R = \begin{bmatrix}
\cos \theta & \sin \theta & 0 \\
-\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}, \\
T = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
T_x & T_y & 1
\end{bmatrix}
\]

So, the composite affine transform is:

\[
SRT = \begin{bmatrix}
S_x \cos \theta & S_x \sin \theta & 0 \\
-S_y \sin \theta & S_y \cos \theta & 0 \\
T_x & T_y & 1
\end{bmatrix} = \begin{bmatrix}
a & b & 0 \\
c & d & 0 \end{bmatrix} = M
\]

\[(4)\]
In the two-dimensional Euclid space, orthogonal transform does not change the vector length and angle. Making \( m = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \), the sufficient condition to keep the metafile shape unchanged is:

\[
\begin{align*}
& a \cdot c + b \cdot d = 0 \\
& \det(m) = \pm 1
\end{align*}
\]  
(5)

If Equation (5) is satisfied, the angle of metafile keeps unchanged. If Equation (6) is satisfied, the length of metafile keeps unchanged. From Equation (4) and (5) we can get:

\[
 a \cdot c + b \cdot d = S_x \cos \theta \cdot ( - S_y \sin \theta ) + S_x \sin \theta \cdot S_y \cos \theta = 0
\]  
(7)

From Equation (7), we can see \( m \) satisfies Equation (5). The angle distortion of metafile in composite affine transform is eliminated and the design requirement of metafile angle unchanged is guaranteed.

### 2.3 Application of dual-buffer technology

When drawing the background of LED pixels on the platform, the content and size of the window changes frequently. The change of window brings the screen flickering. The dual-buffer technology is used to speed up the drawing speed while improving the screen flickering by setting two buffers [9]. The workflow applying dual-buffer technology to solve the problem of screen flickering is shown in Fig. 3.

When drawing background of the platform, in the RAM, a storage area which is the same size of platform window is set to be a container of buffer. The next frame metafile to be shown is painted to the buffer container. When showing metafile on screen, the data is copied directly to the cache. Since process of graphing metafile is on the virtual screen, so screen flicker can be eliminated.

### 3. Debugging and testing of platform

To test the effect of layout platform, a regular and an irregular LED lighting wall are selected as the object, which is shown in Fig. 4.

[Fig. 3 Flowchart of screen refreshing

(a) Regular LED lighting wall  
(b) Irregular LED lighting wall

(a) The logic pixels laid for Fig. 4 (a)  
(b) The logic pixels laid for Fig. 4 (b) ]

When testing the layout effects of regular LED lighting wall, automatic routing mode is used to record the LED logic pixels. The parameter setting interface of automatic routing is shown in Fig. 5, in which the parameters required for auto-routing is set. When laying out for the irregular LED lighting wall, the manual mode of routing is used. The logic diagrams generated by the platform are shown in Fig. 6.
When wiring in the LED specific projects, some physical pixels needs to be adjusted partly, it makes the software platform to be repainted. On the basis of GDI+ technology, the partial edit of metafile is completed. Especially, the shift transform of some pixels is implemented without distortion of picture. The testing results of continuous composite transform without distortion are shown in Fig. 7.

![Fig.7 Screenshots of layout transform](image)

(a) Original layout  (b) composite scaling transformed  (c) composite scaling and shift transformed

4. Conclusions and future work

Combing GDI+ and dual-buffer, the scheme of LED lamps’ layout platform is designed. Through the debugging and testing, it is proved that the developed platform can meet the requirements of engineering applications. The data generated from the layout platform is saved into a file, including the length, width of the layout plane and the two-dimensional coordinates of each logic LED pixel. This is the found for the next step which extracts colour data from video stream and generates the control information stream for LED lighting system. In a more complex heterogeneous system, the modelling and laying out in the three-dimensional physical background is looking forward.

Acknowledgements

The work is supported by China Postdoctoral Science Foundation (2011M500130) and Zhongshan Science and Technology Research Project (20114A159).

References

[9] Xia Min; Wu Zhonghai; Yang Yahui, Sending Rate Control Method Based on Double Buffer, 2010, 36(5):119-121