目　　录

1 需求分析

1.1功能需求分析

1) 闪屏动画：登录窗口前展示闪屏动画，等待程序加载。
2) 登录及注册：用户登录时与用户信息存储文件中数据进行对比，存在方可登录；用户注册时将用户信息按照规则写入用户信息存储文件中，并且可以进行有效读取。
3) 新建，打开：进入主界面后默认无法绘制，工具栏不可用，单击新建或打开文件按钮时可以进行相应新建绘图并将窗体背景改为白色或打开电脑中存在的图元或图片操作，新建或打开图元或图片后方可进行选择工具栏中工具进行矢量图绘制。
4) 矢量图绘制：选择工具栏中相应工具，可以进行直线，矩形，椭圆，折线，（不规则）多边形等图形的绘制。图形绘制过程随刷新不断显示（鼠标移动时不断显示绘制的图形）。大小可根据窗体大小改变而缩放。
5) 保存：将当前绘图以图元或图片格式保存至电脑中。
6) 画笔颜色，粗细选择：单击画笔颜色，画笔粗细按钮可以相应更改画笔的颜色或粗细。
7) 帮助：单击帮助按钮，可以唤起默认浏览器跳转至帮助文档页面
8) 实时时间：主界面右下方实时显示当前时间。
9) 退出：单击退出或程序窗口右上角“X”可以退出程序。

1.2 性能需求分析

1．硬件环境

- 处理器：Inter CR300 或是更高。
- 内存：128MB（建议 196MB）。
- 硬盘空间：20MB。

2．软件环境

- 操作系统: Windows 98/ XP 或是 Windows 2000/Windows NT Server 4.0。

1.3 系统总体结构设计

对简易矢量图绘制系统的设计，主要分为以下三个模块：

闪屏窗体：闪屏图片，欢迎提示，滚动进度条

登录窗体：登录，注册

主窗体：文件新建或打开，文件保存，图形绘制，画笔选择，帮助文档，实时时间，程序退出

根据需求分析的结果, 总体结构如图 1 所示。

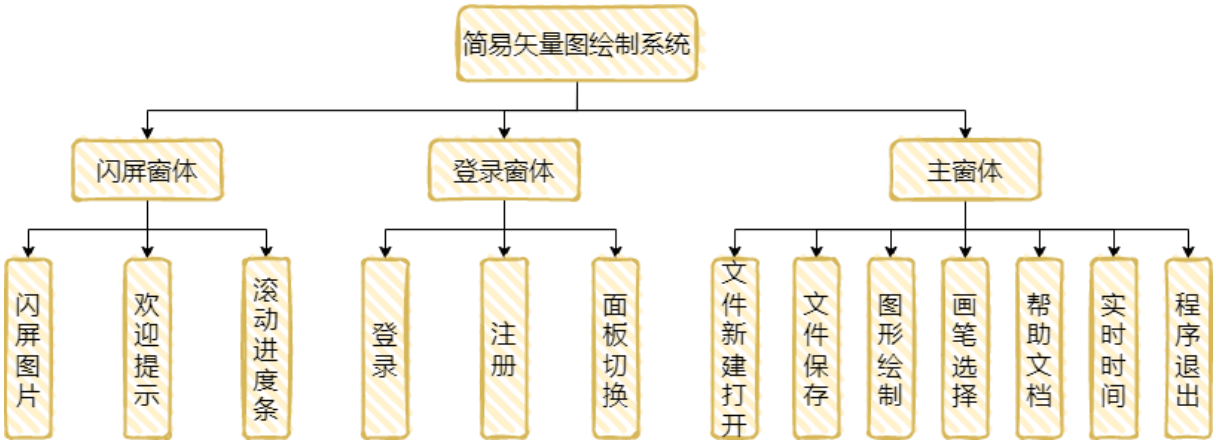

图 1 系统总体结构

2 详细设计

2.1 界面设计

1） 闪屏动画：欢迎提示采用 Label 实现，滚动进度条采用 ProgressBar 实现。

2） 登录及注册：登陆及注册窗体中，用户名及密码标签采用 Label 实现，用户名及密码文本框采用 TextBox 实现，确定及重置按钮采用 Button 实现，“没有账号？去注册”及“已有账号？去登录”可点击标签采用 LinkedLabel 实现，登录及注册面板放在 Panel 中，方便实现面板切换。

3） 新建，打开：新建，打开按钮采用 ToolStripMenuItem 实现，打开面板采用 openFileDialog 实现。

4） 矢量图绘制：工具栏采用 ToolStrip 实现，工具栏中各工具采用 ToolStripButton 实现。

5） 保存：保存按钮采用 ToolStripMenuItem 实现，打开面板采用 saveFileDialog 实现。

6） 画笔颜色，粗细选择：画笔颜色，粗细选择按钮采用 ToolStripMenuItem 实现，颜色选择面板采用 colorDialog 实现，画笔粗细选择采用 ToolStripMenuItem 实现。

7） 帮助：帮助按钮采用 ToolStripMenuItem 实现。

8） 实时时间：实时时间显示采用 Label 实现。

9） 退出：退出按钮采用 ToolStripMenuItem 实现。

2.2 功能实现设计

该简易矢量图绘制系统主要包含三个界面，即闪屏界面，登陆界面，主界面

（1） 闪屏界面

相关技术使用方面，闪屏界面使用了 System.Windows.Forms.Timer 技术，用于实现滚动进度条随闪屏时间的实时更新。用到了 Timer_Tick 方法，通过在规定间隔后设置进度条的值，实时更新进度条及进度条加载提示， 同时检查当进度条值达到 100%后关闭当前窗口。

控件使用方面，闪屏界面使用 ProgressBar 进度条控件，用于实现表示程序等待加载的滚动进度条，同时使用 Label 控件，用于表示进度条加载进度即欢迎提示。

功能实现方面，如图 2 所示，通过 Timer 及 ProgressBar 控件实现了进度条的实时更新表示了程序等待加载，同时通过 Label 显示了进度条加载进度。

```
    timer = new System.Windows.Forms.Timer();
    timer.Tick += Timer_Tick;
    timer.Interval = interval;
    timer.Start();
}

private void Timer_Tick(object sender, EventArgs args)
{
    // 更新进度条
    currentProgress++;
    progressBar.Value = currentProgress;
    label.Text = $"欢迎使用“易失绘”！正在加载程序... {currentProgress}%";

    // 检查是否达到100%
    if (currentProgress >= 100)
    {
        timer.Stop();
        Close();
    }
}
```

图 2 闪屏界面实现

（2） 登录界面

相关技术使用方面，登录界面使用了 System.IO.File 类相关技术，用于实现文件的读写及存在检查等，用到了 File 类的 Exists 方法及 Create 方法，以及 WriteLine 方法和 ReadLine 方法等。实现了登录及注册相关操作。

控件使用方面，登录和注册界面分别使用 Panel 控件实现，登录和注册界面的用户名及密码标签使用 Label 控件，用户名及密码文本框使用 TextBox 控件实现，确定和重置的按钮使用 Button 控件实现，选择切换登录及注册界面的可点击标签使用 LinkedLabel 实现，点击即可通过设置 Panel 的 Visible 属性实现。

功能实现方面，如图 3，4，5 所示，通过 File 类相关方法实现了文件的读写及存在检查，通过 LinkedLabel 及 Panel 的 Visible 属性设置实现了注册及登录界面的切换。

```
// 注册窗口点击事件
private void okButton2_Click(object sender, EventArgs e)
{
    // 检查文件是否存在，不存在则创建
    if (!File.Exists("user_info.txt"))
    {
        File.Create("user_info.txt").Close();
    }
    // 写入用户信息到文件中
    using (StreamWriter sw = new StreamWriter("user_info.txt", true))
    {
        sw.WriteLine($"{Username2}:{Password2}");
        MessageBox.Show("注册成功。", "注册提示", MessageBoxButtons.OK);
    }
}
```

图 3 注册相关技术实现

```
//登录比对
private bool Validate(string username, string password)
{
    // 检查文件是否存在，不存在则说明没有注册过用户
    if (!File.Exists("user_info.txt"))
    {
        return false;
    }

    // 读取文件中的用户信息，并逐行比对
    using (StreamReader sr = new StreamReader("user_info.txt"))
    {
        string line;
        while ((line = sr.ReadLine()) != null)
        {
            string[] userInfo = line.Split(':');
            if (userInfo[0] == username && userInfo[1] == password)
            {
                return true;
            }
        }
    }
    return false;
}
```

图 4 登录相关技术

```
private void resetButton_Click(object sender, EventArgs e)
{
    usernameTextBox.Text = "";
    passwordTextBox.Text = "";
}
private void linkedLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    panel1.Visible = false;
    panel2.Visible = true;
}
```

图 5 登陆注册界面切换实现

（3） 主界面

相关技术使用方面，主界面使用了 System.Drawing 相关方法实现了图形的绘制，使用 System.Threading 相关方法实现了使用多线程技术显示主窗体右下方实时时间，使用 System.Diagnostics 相关方法实现了启动浏览器进程打开帮助文档的操作，使用了 System.Drawing.Imaging 相关方法实现了绘图保存成图元等格式的操作，使用了 System.Collections.Generic 相关方法实现了 Point 列表的使用。

控件使用方面，主界面使用了 MenuStrip 菜单栏控件，以及 ToolStrip 工具栏控件，菜单栏及工具栏中每项通过 ToolStripMenuItem 及 ToolStripButton 实现，点击即可进行相关操作。界面右下角时间的显示采用 Label 控件实现。

功能实现方面，如图 6，7，8，9，10 所示，主界面通过 saveFileDialog 及 openFileDialog 相关方法实现了文件的保存及打开，通过 colorDialog 实现了颜色的选择，通过 Process 相关方法实现了启动浏览器进程，打开帮助文档的操作，通过 Drawing 类等实现了图形的绘制，以及将绘图保存成图元等相关格式。通过 Threading 类相关方法实现了应用多线程进行实时时间的显示。

```
private void 打开ToolStripMenuItem_Click(object sender, EventArgs e)
{
    openFileDialog1.Filter = "bmp|*bmp|wmf|*wmf|ico|*ico|cur|*cur|jpg|*jpg";
    openFileDialog1.Multiselect = false;
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        this.Text = "MyDraw\t" + openFileDialog1.FileName;//打开的窗体名字
        editFileName = openFileDialog1.FileName;
        theImage = Image.FromFile(openFileDialog1.FileName);
        Graphics g = this.CreateGraphics();
        g.DrawImage(theImage, this.ClientRectangle);
        ig = Graphics.FromImage(theImage);
        toolStrip1.Enabled = true;
    }
}
```

图 6 主界面打开功能实现

```
private void 新建ToolStripMenuItem_Click(object sender, EventArgs e)
{
    theImage = new Bitmap(this.ClientRectangle.Width, this.ClientRectangle.Height);
    editFileName = "新建绘图";
    this.Text = "MyDraw\t" + editFileName;
    ig = Graphics.FromImage(theImage);
    ig.Clear(backcolor);
    toolStrip1.Enabled = true;
}

private void 保存ToolStripMenuItem_Click(object sender, EventArgs e)
{
    saveFileDialog1.Filter = "图像(*.bmp)|*.bmp";
    saveFileDialog1.FileName = editFileName;
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        theImage.Save(saveFileDialog1.FileName, ImageFormat.Bmp);
        this.Text = "MyDraw\t" + saveFileDialog1.FileName;
        editFileName = saveFileDialog1.FileName;
    }
}
```

图 7 主界面新建保存实现

```
private void 帮助ToolStripMenuItem_Click(object sender, EventArgs e)
{
    // 指定要打开的 URL
    string url = "https://docs.microsoft.com/en-us/visualstudio/?view=vs-2022";

    try
    {
        // 启动浏览器进程，并打开指定的 URL
        Process.Start(url);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

图 8 主界面帮助文档跳转实现

```
switch (drawTool)
{
    case drawTools.Line:
        ig.DrawLine(new Pen(forecolor, fontsize), startPoint, new Point(e.X, e.Y));
        break;
    case drawTools.Rectangle:
        ig.DrawRectangle(new Pen(forecolor, fontsize), startPoint.X, startPoint.Y, e.X - startPoint.X, e.Y - startPoint.Y);
        break;
    case drawTools.Ellipse:
        ig.DrawEllipse(new Pen(forecolor, fontsize), startPoint.X, startPoint.Y, e.X - startPoint.X, e.Y - startPoint.Y);
        break;
    case drawTools.BLine:
        if (points.Count > 1)
        {
            for (int i = 0; i < points.Count - 1; i++)
            {
                ig.DrawLine(new Pen(forecolor, fontsize), points[i], points[i + 1]);
            }
        }
        break;
    case drawTools.Polygon:
        if (points.Count > 1)
        {
            for (int i = 0; i < points.Count - 1; i++)
            {
                ig.DrawLine(new Pen(forecolor, fontsize), points[i], points[i + 1]);
            }
            ig.DrawLine(new Pen(forecolor, fontsize), points[points.Count - 1], points[0]);//画首尾连接线
        }

        break;
}
```

图 9 主界面绘制实现

```
// 更新系统时间的方法
private void UpdateClock()
{
    while (true)
    {
        // 获取当前系统时间
        DateTime now = DateTime.Now;

        // 在主线程中更新UI
        this.Invoke((MethodInvoker)delegate
        {
            // 在状态栏中显示系统时间
            timeLabel.Text = now.ToString("yyyy-MM-dd HH:mm:ss");
        });

        Thread.Sleep(1000); // 休眠1秒
    }
}
```

图 10 主界面时间显示实现

3 运行效果说明

1、首先给出的是闪屏界面部分的运行情况，如图 11 所示。

图 11 闪屏界面图

用户运行程序后，首先展示闪屏界面，表示程序正在加载，当闪屏界面下方加载条满时，程序加载成功，展示登录界面。

2、登录界面运行情况如下，如图 12，13 所示。

图 12 登录界面图

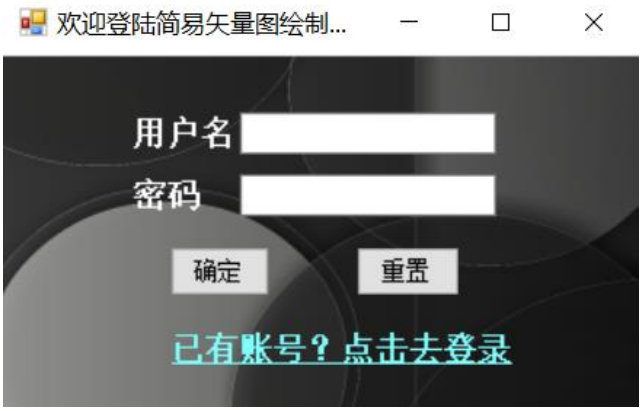

图 13 注册界面图

当程序闪屏界面结束后，展示登录界面，如图所示，用户可在用户名和密码相应文本框输入自己的用户名和密码，点击确定按钮进行登录，点击重置按钮可以重置用户名和密码文本框的值。如果用户还没有账号，可以点击下方链接跳转至注册界面，如图所示，在注册界面输入相应用户名和密码，点击确定按钮进行注册，弹窗提示注册成功后，用户可点击下方链接跳转登录界面进行登录。

3、主界面运行情况如下，如图 14，15 所示。

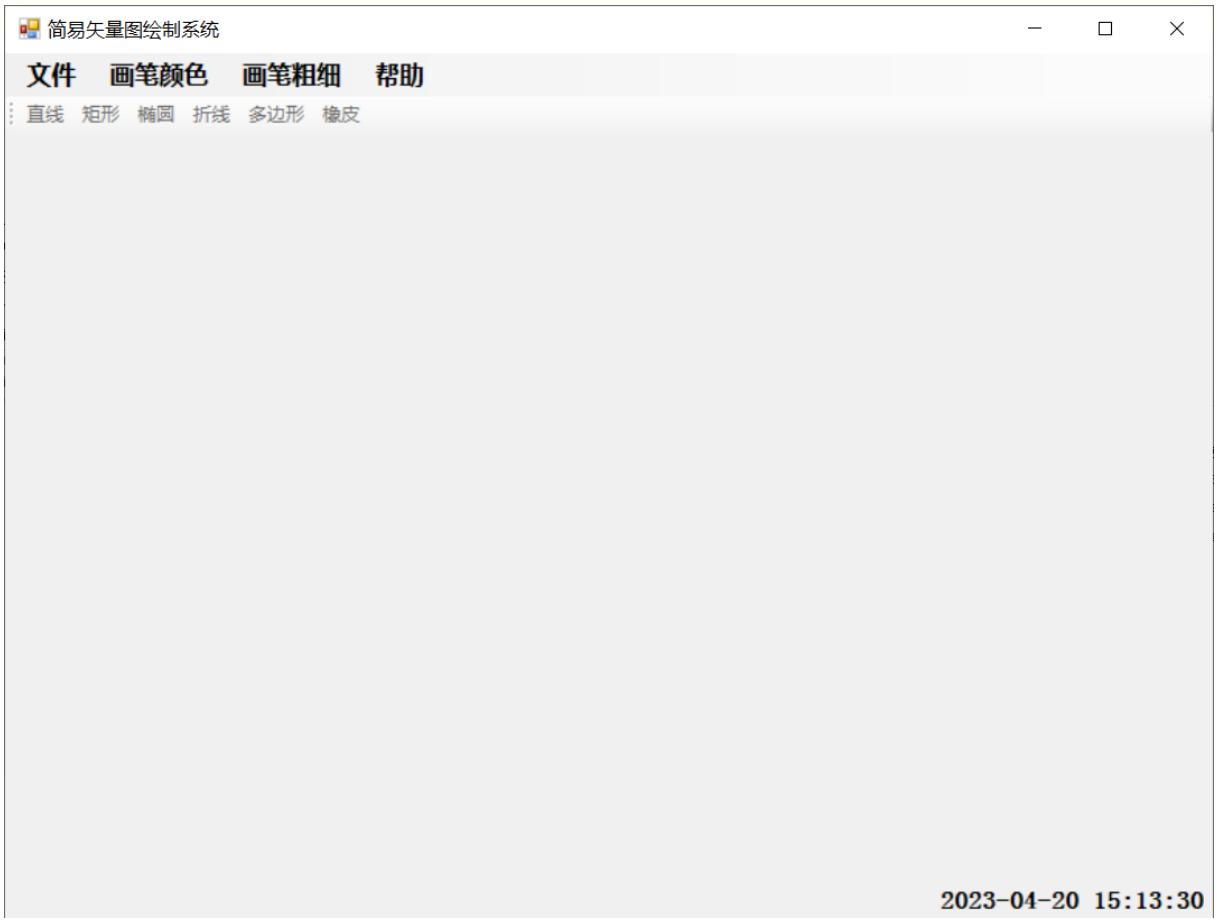

图 14 主界面默认界面图

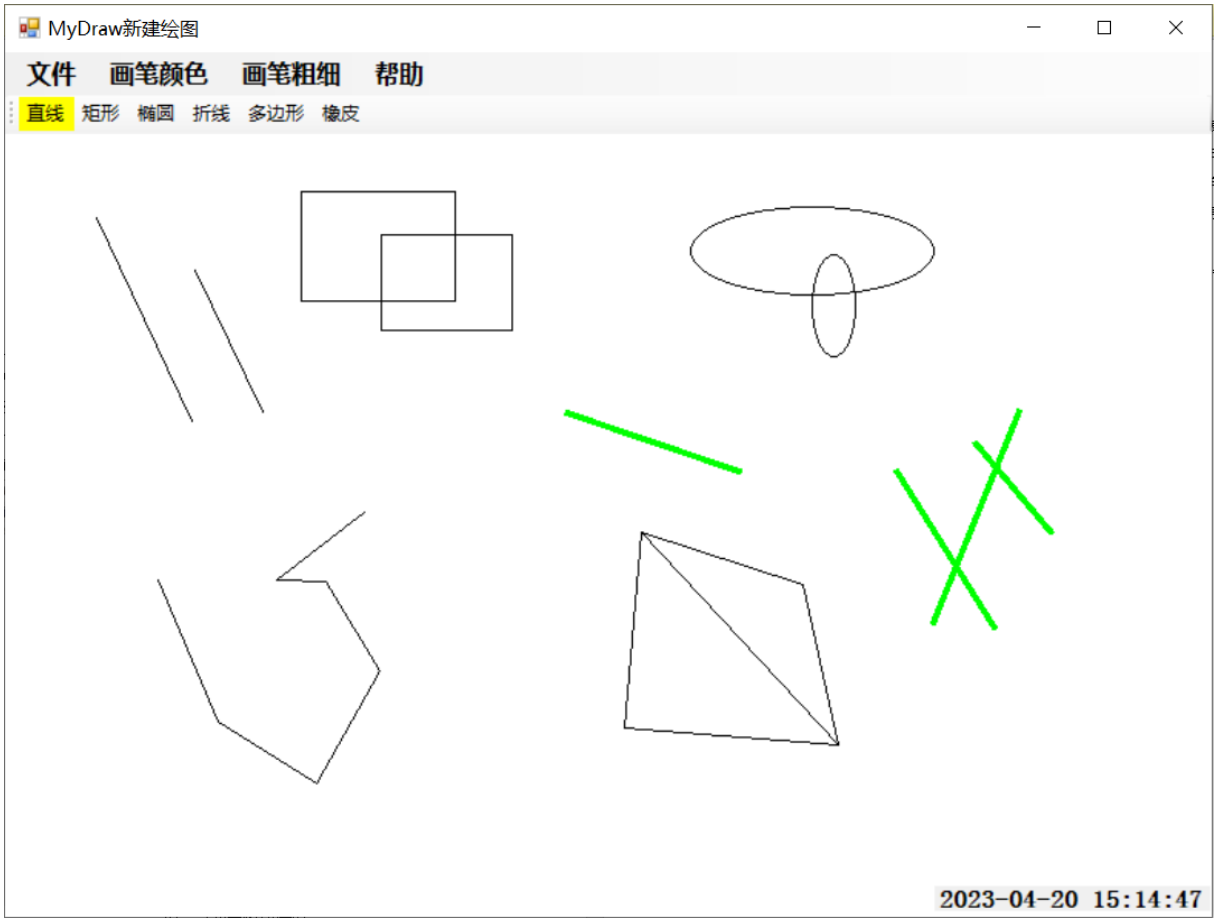

图 15 主界面绘制界面图

当用户成功登录简易矢量图绘制系统后，进入主界面，主界面默认界面工具栏无法使用，如图所示，当用户点击菜单栏中文件，选择新建或打开后，工具栏方可解锁使用，此时用户可以在工具栏选择图形，并在菜单栏选择画笔颜色和粗细，在下方画板进行相应绘制。当用户绘制完成后，点击文件下拉选项的保存按钮，即可将当前绘图保存至电脑中。同时主界面右下方有当前系统时间显示，点击退出按钮即可退出程序。点击菜单栏中帮助按钮程序会唤起默认浏览器打开帮助文档界面，供用户进行查看。

4 实现难点及解决方案

4.1 绘制过程闪烁

在实际的绘制过程当中，在鼠标移动时绘制过程能够随刷新不断显示，即鼠标移动时不断显示绘制的图形，但也因此，可能因为画面的重绘次数较多，导致出现了画面闪烁的现象。这样不仅会影响用户的使用体验，还更可能导致绘图效果的不理想。

我使用了以下几种方法结合使用来解决这个难点问题：

（1）使用双缓冲技术

在绘制图形时，使用双缓冲技术可以有效地减少画面闪烁的现象。具体实现方法是，在绘制之前，先在内存中创建一个与画布大小相同的位图，并在位图上进行绘制，绘制完成后再将位图显示在屏幕上。这样可以避免图形在显示时出现闪烁。

（2）调整绘制顺序

在绘制图形时，可以按照从后往前的顺序绘制，即先绘制在底层的图形，再绘制在上层的图形。这样可以避免在绘制过程中出现前景和背景的混合，从而减少画面闪烁的现象。

（3）减少无效的重绘

在绘制过程中，如果只重绘发生变化的部分，可以有效地减少无效的重绘，从而减少画面闪烁的现象。可以使用刷新区域的方法，即只重绘发生变化的区域，避免整个画面的重绘。

4.2 文件的读写与存储

在本简易矢量图绘制系统中，需要支持将绘制的图形以某种格式保存到文件中，并能够在需要时读取该文件中的数据，同时也要支持读取电脑中已有的某些格式的文件。这就需要涉及考虑到文件以何种格式存储，如何实现读写操作等问题。

为解决该问题，我使用了 C# WindowsForm 自带的 saveFileDialog 及 openFileDialog 控件来进行绘图文件的保存及打开。可以将绘图文件保存为 bmp，jpg，png 等格式。同时支持读取 bmp，jpg，png，ico 等多种格式的文件。

在用户信息的存储方面，使用了文件操作相关的 API 将用户登录及注册信息保存到文件中，并实现了登录注册时的读取和存入。在用户进行注册时，使用 System.IO.File 类中的 WriteLine 方法将信息写入至文本文件中。在用户登录时，使用 System.IO.File 类中的 ReadLine 方法读取文本文

件中的数据。同时还使用了 System.IO.File 类中的 Exits 方法，判断文件在当前目录下是否存在。

4.3 折线及多边形的绘制

在绘制功能实现中，折线及多边形的绘制也是一大难点。当用户在工具栏中选择折线或多边形后，需要考虑画笔的绘制逻辑如何实现。如点的选取问题：在绘制折线或多边形时，需要用户自行选择点的位置以确定线条的走向。又如绘制连接处的处理：在绘制折线或多边形时，需要考虑到连接处的处理，保证线条的顺畅和连贯。

为解决该问题，我设置 Point[]数组，当用户在工具栏选择折线或多边形后，当用户每使用鼠标左键在画板上单击一次，记录下该点的位置，并存储到 Point[]数组中，当 Point[]数组的长度大于等于 2 时，则开始进行折线的绘制，每次绘制以此次记录的上一个点为起始点，以此次记录的点为终点绘制直线，这样就实现了折线及多边形的绘制。同时当面板刷新时，清空 Point[]数组，保证下次绘制时与上次绘制折线无关，不会对上次绘制的折线及多边形造成影响。