2023

AnimSprite

软件构造基础大作业

汇报人：薛瑞年
小组成员：薛瑞年、王少鹏、罗昱奇、曹祎琨

目录

Content

PART 01

选题动机

Background

选题动机

在计算机科学技术日渐发展的今天，我们使用电脑的频率越越发增加，我们通过游戏和各种渠道接触到了自己喜欢的角色，因此我们想要把它应用于我们每日电脑使用的陪伴当中。

基于此，我们小组设计了一个基于桌面精灵功能的一个桌面助手，希望在方便日常生活的同时，也能够美观，简洁，精准地实现相应的每日需求。

PART 02

分工介绍

Introduction of work division

分工介绍

01 薛瑞年

前端设计，事件处理

前端设计，设计了相关的交互模式和运行的接口。实现桌面精灵状态和相关功能要求的对接实施。具体体现在事件的产生与应答。后续优化了UI的表现形式，使得项目的外观简洁美观。

分工介绍

02 王少鹏

后端处理，对外接口设计，事件处理

负责后端框架设计和处理，包括备忘录提醒、CPU和内存监测以及更换壁纸功能。编写后端接口，并利用反射技术实现功能的扩展性，只需修改配置文件而不是主体代码，确保系统的开闭性良好。

分工介绍

03 罗昱奇

UI优化

分工介绍

04 曹祎琨

资料收集、UI优化、开题报告整理

PART 03

软件功能

Main funtions

软件功能

功能概述

作为一个基本的桌面精灵，首先要求实现基础的移动动作和相关的互动，包括角色的待机和在屏幕上的移动。以及与桌面相关控件的互动。

其次，作为一个助手，我们为桌面精灵还添加了与电脑控制提醒相关的基础功能，比如CPU温度，内存占用等。其中CPU温度和内存占用会以十秒钟一次的频率检查是否温度过高或占用过大。若占用过大会发出警报提醒用户及时处理。

除此之外，我们还添加了备忘录和提醒事项功能。

我们的所有功能几乎都以与角色互动的形式进行，而并非简单的以按键的形式呈现。这样的呈现效果会更加的生动。

PART 04

技术路线

Key points of implementation

实现难点

窗口界面的实现

如何利用WPF实现基本的异形窗口，并且能够与用户在适当的时候进行互动。

接口的调用

要实现内存，CPU调用等功能，需要与Windows自带的相关应用API进行交互

对话界面的互动

利用对话界面互动而非实体的按钮，需要前端对相关的事件进行识别，并且发送到后端进行交互。

事件的响应和产生

前端需要产生需求，发送事件到后端，后端同样会监控相应的电脑状态，并将异常的事件发送到前端，需要两者进行事件的交互与监控。

UI的优化

UI不应该简单的调用WPF其中的相关的控件，因为这样会比较的丑陋，并且缺乏美观度。

前后端架构搭建

项目前后端分离，同时桌面精灵需要不断进行交互和事件响应，功能较为琐碎，需要搭建出合适的架构确保项目良好的完整性和可拓展性。

实现细节

反射技术实现后端服务集成

```
public string getResponds(string s)
{
    if (s == null) return " ";
    string[] keys = ConfigurationManager.AppSettings.AllKeys;
    foreach (string key in keys)
    {
        if (s.Contains(key))
        {
            // return ConfigurationManager.AppSettings[key];
            MethodInfo methodInfo = this.GetType().GetMethod(ConfigurationManager.AppSettings[key]);
            if (key.Contains("添加备忘录")|| key.Contains("完成备忘录")|| key.Contains("删除备忘录"))
            {
                try
                {
                    string[] parts = s.Split(' : ');
                    request = ConvertChineseNumberToArabic(parts[1].Trim());
                }
                catch
                {
                    return " ";
                }
            }
            if (methodInfo != null) return methodInfo.Invoke(this, null).ToString();
            else return s;
        }
    }
    return s;
}
```

实现细节

配置文件

```
<appSettings>
    <add key = "内存" value="getMemory" />
    <add key = "cpu" value="getcpu" />
    <add key = "壁纸" value="setNextWallpaperGallerys" />
    <add key = "添加备忘录" value="m_add" />
    <add key = "删除备忘录" value="m_del" />
    <add key = "完成备忘录" value="m_change" />
    <add key = "查询备忘录" value="m_find" />
    <add key = "清空备忘录" value="m_clear" />
</appSettings>
```

实现细节

备忘录时间自然语言处理（正则表达式）

```
private static DateTime GetNextDateTime(DateTime now, DayOfWeek targetDayOfWeek, int daysToAdd)...
Dictionary<string, int> dayModifiers = new Dictionary<string, int>...;

Dictionary<string, DayOfWeek> dayOfWeekPhrases = new Dictionary<string, DayOfWeek>...;
1 个引用
public DateTime? getSetTime(string input)
{
    DateTime now = DateTime.Now;
    DayOfWeek currentDayOfWeek = now.DayOfWeek;
    myevent = " ";
    string pattern = @"(?i)(\b明天|\b后天|\b大后天)?(\b本周|\b下周)?([一二三四五六日周])?(早上|上午|下午|晚上)?(\d+)点(\d+)?(分(钟)?)?([\u4E00-\u9FFF]+)?";
    Match match = Regex.Match(input, pattern);
    if (match.Success)...
    match = Regex.Match(input, @"(\d+)(个)?(半)?((小时)?|(分钟)?|(秒钟)?)(\d+)?((小时)?|(分钟)?|(秒(钟)?)?)后([\u4E00-\u9FFF]+)?");
    if (match.Success)...
    return null;
}
```

实现细节

电脑状态监控

```
public void monitorStatus(object sender, EventArgs e)
{
    StringBuilder sb = new StringBuilder();
    if (_availableMemory < 5)
    {
        sb.Append("当前总内存：").Append(totalMemory).Append("G\n").Append("可用内存：").Append(_availableMemory).Append("G，内存使用有些过高\n");
    }
    if (_temperature > 70)
    {
        sb.Append("当前cpu温度：").Append(_temperature).Append("℃，温度有些过高");
    }
    if(sb.Length > 0)
    throwRequest(sb.ToString());
}
```

实现细节

备忘录提醒事件监控

```
public void monitorEvents(object sender, EventArgs e)
{
    for(int i = 0; i < m.getTime.Count; i++)
    {
        if (DateTime.Now > m.getTime[i])
        {
            throwEvents(m.getEvent(i));
        }
    }
}
```

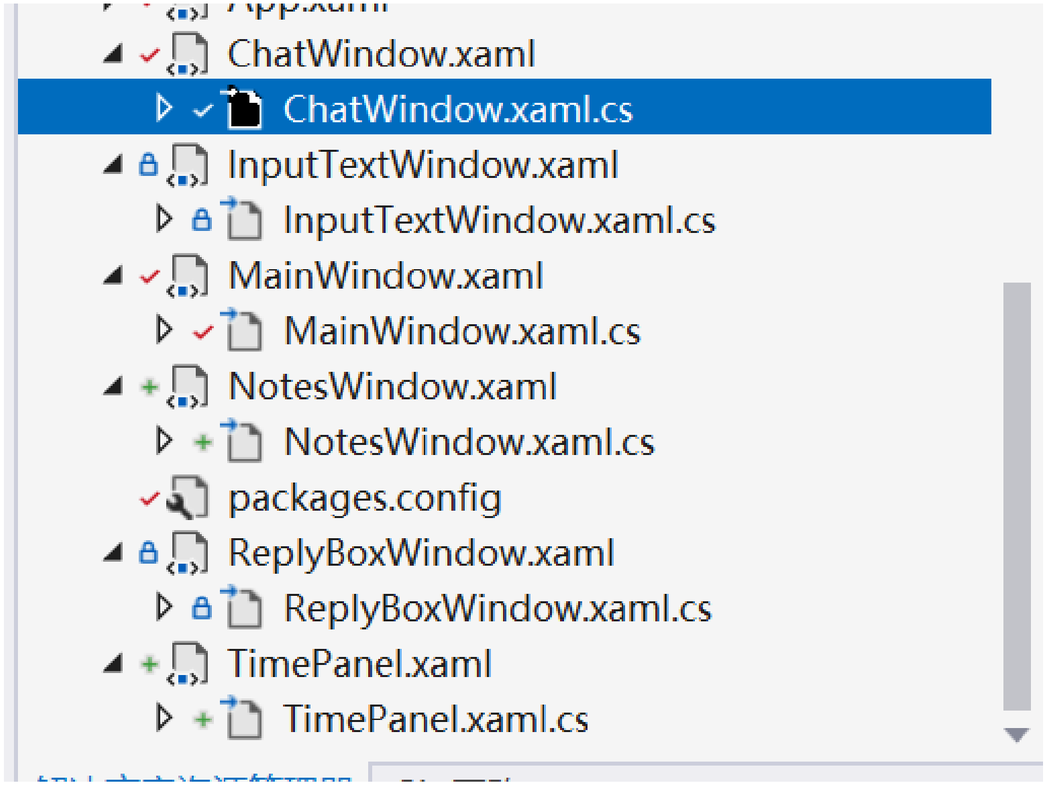
实现细节

监控实现

```
c = new Controller();
updataStatusTimer =  new Timer(c.setInfoEvent, null, 0, 1000);
c.throwRequest += getRequest;
c.throwEvents += getEventRepond;
monitorStatusTimer = new DispatcherTimer();
monitorStatusTimer.Interval= TimeSpan.FromSeconds(50);
monitorStatusTimer.Tick += c.monitorStatus;
monitorStatusTimer.Tick +=c.monitorEvents;
monitorStatusTimer.Start();
```

实现难点

窗口界面的实现



从上到下分别实现了聊天对话框、主要的角色框、备忘录显示界面、消息警告界面、和时间窗口

实现难点

角色移动的实现

```
<Window x:Class="TestWPF.MainWindow"
        xmlns="http://schemas.microsoft.com/win
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:gif="http://wpfanimatedgif.codeplex.com"
        xmlns:local="clr-namespace:TestWPF"
        mc:Ignorable="d"
        SizeToContent="WidthAndHeight"
        SizeChanged="Window_SizeChanged"
        Title="DeskSparite"
            Height="100" Width="100"
            WindowStyle="None"
            AllowsTransparency="True"
            Background="Transparent"
            Topmost="True"
            WindowStartupLocation="Manual" Left="0" Top="0"
            ResizeMode="CanResizeWithGrip"
```

```
//loadImage
private void LoadGifImage(string gifMode)
{
    var gifPath = System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, $"../../Resources/{gifMode}.gif");
    var gifImage = new BitmapImage(new Uri(gifPath));
    ImageBehavior.SetAnimatedSource(GifImage, gifImage);
    GifImage.Width = this.Width;
    GifImage.Height = this.Height;
}
```

```
private void MoveTimer_Tick(object sender, EventArgs e)
{
    // Check if left mouse button is pressed
    if (Mouse.LeftButton == MouseButtonState.Pressed)
    {
        // Stop moving and clear moveTimer
        isMoving = false;
        moveTimer.Stop();
        moveTimer = new DispatcherTimer();
        //moveTimer.Start();
        RandomStaying();
        return;
    }
    if (isMoving)
    {
        this.Left += deltaX;
        this.Top += deltaY;

        // Check for change in direction and update GIF
        if (deltaX < 0 && moveGif != "Walk_left")
        {
            moveGif = "Walk_left";
            LoadGifImage(moveGif);
        }
        else if (deltaX > 0 && moveGif != "Walk_right")
        {
            moveGif = "Walk_right";
            LoadGifImage(moveGif);
        }

        // Check for collision with screen edges
        if (this.Left < 0)
        {
            this.Left = 0;
            deltaX = -deltaX;
```

实现难点

前后端交互的实现——对话

```
public void ReplyMessage(string s)
{
    string messageText = s;//这里需要修改！！！！！
    NewMessageBubble(false, messageText);
}
string added="";
private void SendMessage(object sender, RoutedEventArgs e)
{
    string messageText = "";
    messageText = added+messageInput.Text;//这个是message，可以被读取

    NewMessageBubble(true, messageText);
    OnMessageReceived(messageText,2);
    // Clear the input TextBox
    messageInput.Text = "";
    added = "";

}
private void SendMessage()
{
    string messageText = "";
    messageText = added + messageInput.Text;//这个是message，可以被读取

    NewMessageBubble(true, messageText);
    OnMessageReceived(messageText, 2);
    // Clear the input TextBox
    messageInput.Text = "";
    added = "";

}
```

ChatWindow

```
requestWindow = new ChatWindow();
requestWindow.Owner = this;
requestWindow.Show();
requestWindow.OnMessageReceived += ProcessRequests;
requestWindow.closed += ShowChatButton;
//获取文本
return requestWindow.SendText;
```

MainWindow

实现难点

对话框和聊天框

```
private void NewMessageBubble(bool isSendMessage,string messageText)
{

    // Save the message to the file
    if (isSendMessage)
    {
        if(messageText.Contains("bye")||messageText.Contains("再见"))
        {
            closed();
            Close();
            return;
        }
        SendText = messageText;
        ChatHistoryFile.Save("[Send]" + messageText);
    }
    else
    {
        ChatHistoryFile.Save("[Reply]" + messageText);
    }

    // Create a new Grid element for the message bubble
    Grid messageBubble = new Grid();
    messageBubble.Margin = new Thickness(0, 0, 0, 10);

    // Create a new Ellipse element for the avatar circle
    Ellipse avatarCircle = new Ellipse();
    avatarCircle.Width = 40;
    avatarCircle.Height = 40;

    // Create a new Border element for the message text
    Border messageTextBorder = new Border();
    messageTextBorder.CornerRadius = new CornerRadius(10);
    messageTextBorder.Padding = new Thickness(10);

    if (isSendMessage)
    {
        avatarCircle.Fill = Brushes.LightBlue;
        messageTextBorder.Background = Brushes.LightBlue;
    }
```

实现难点

快捷方便的交互

```
<!-- Additional buttons -->
<Grid Grid.Row="2" Margin="10">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>

    <Button Grid.Column="0" Content="DoneM" Width="80" Click="Button_Click_4" hc:BorderElement.CornerRadius="15"/>
    <Button Grid.Column="1" Content="DeleteM" Width="80" Click="Button_Click_3" hc:BorderElement.CornerRadius="15"/>
    <Button Grid.Column="2" Content="Addm" Width="80" Click="Button_Click_2"  hc:BorderElement.CornerRadius="15"/>
</Grid>

<!-- Additional buttons -->
<Grid Grid.Row="3" Margin="10">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>

    <Button Grid.Column="0" Content="GetM" Width="80" Click="Button_Click_1"  hc:BorderElement.CornerRadius="15" />
    <Button Grid.Column="1" Content="ClearM" Width="80" Click="Button_Click"  hc:BorderElement.CornerRadius="15"/>
    <Button Grid.Column="2" Width="80" Content="Clear" Click="ClearMessages_Click"  hc:BorderElement.CornerRadius="15" />
</Grid>
```

前端实现美化用到的库：

WPF、AnimeGif、handycontrol

PART 05

效果展示

View of project

效果展示

基本互动

演示文稿是一种交流工具，可以用作演示、讲座、演讲、报告等。

状态监控

演示文稿是一种交流工具，可以用作演示、讲座、演讲、报告等。

备忘

演示文稿是一种交流工具，可以用作演示、讲座、演讲、报告等。

效果展示

视频演示

PART 06

项目总结

Summary

项目总结

项目亮点

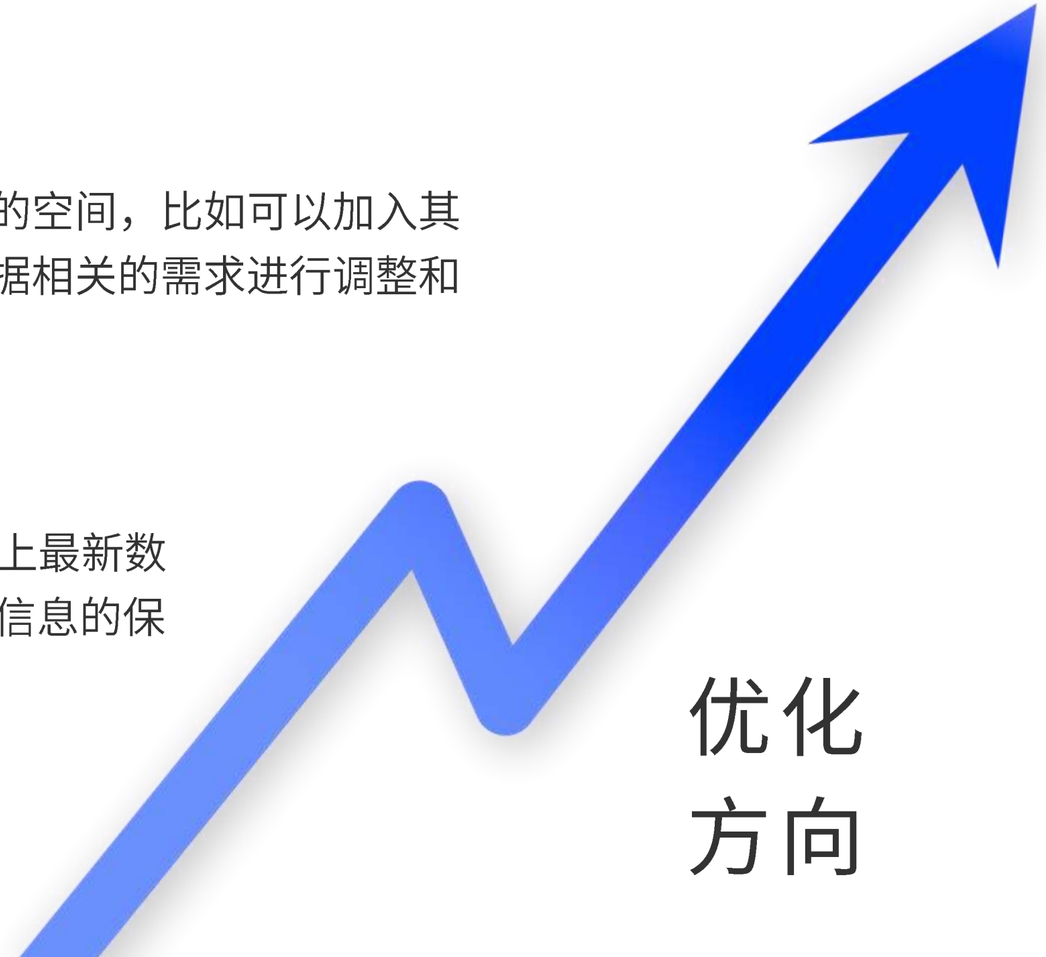
1. 实现了有趣的互动功能
2. 基于对话响应用户需求
3. 界面简洁美观
4. 实现了事件的处理和响应
5. 提供了抽象的接口和配置文件，方便随时插入新插件

项目总结

1 相关的扩展功能，比如基于chatGPT的API的互动和交流功能，或者基于LangChain的交互功能未能实现，相关的交互显得略为笨拙和迟缓。

2 UI并不是十分的精简和美观，还有提升的空间，比如可以加入其他的控件，使得整个软件的UI也可以根据相关的需求进行调整和更换，类似于更换皮肤。

3 可以提供一些外部接口的示例，并付诸实践，比如Arxiv上最新数据的调取（结合网络爬虫），或者加入数据库对于相关信息的保存功能，而非将大多数信息存储在简单的txt文件中。



优化方向

2023

谢谢观看

Thank You for watching