用 C# 实现文本朗读和语音识别功能

彭　腾　王小红

摘　要：通过一个实例，介绍在 C# 中如何实现文本朗读和语音识别功能，对于设计人性化的界面提供了一种比较好的参考。

关键词：语音识别；文本朗读

1　引言

随着语音合成与识别技术的进步，开发人机界面和计算机辅助外语学习软件越来越依赖于这两项技术。在.Net Framework 3.5 中，可以使用命名空间：System.Speech 这个集合的类和方法来实现文本朗读与语音识别功能。

2　概述

2.1　语音开发平台配置

由于 Windows XP 只整合了语音合成引擎 (TTS)，不能朗读中文和识别语音并且只提供了 Microsoft Sam 朗读角色，所以开发能朗读和识别中、英文，需安装以下软件:

(1) 安装 Microsoft Speech SDK5.1 (68MB)

http://download.microsoft.com/download/speechSDK/SDK/5.1/WXP/EN-US/Speechsdk51.exe

(2) 安装 Microsoft Speech SDK5.1 Language Pack (81.5MB)

http://download.microsoft.com/download/speechSDK/SDK/5.1/WXP/EN-US/Speechsdk51LangPack.exe

2.2　System.Speech 命名空间集合

System.Speech 命名空间集合包含两个命名空间：Synthesis 和 Recognition。其中 Synthesizer 命名空间提供的 SpeechSynthesizer 类，可以轻松实现文本朗读功能；Recognition 空间提供了语言识别类 SpeechRecognizer。

图 1

2.3　添加引用

要使用 SpeechSynthesizerhe、SpeechRecognizer 这两个类需添加引用。新建一个 Windows 窗体应用程序，点击“项目”菜单下的“添加引用”子菜单，在“.Net 选项”中选择 System.Speech 组件名称，点击确定按钮，如图 1 所示。

3　实现文本朗读功能

3.1　二次封装 SpeechSynthesizer 类

程序采用单例设计模式对 SpeechSynthesizer 类进行二次封装，单例模式确保 SpeechSynthesizer 只有一个实例，这样可以大幅度减少程序占用的内存 。具体做法是新建 Talker 类，在类内部声明两私有变量 syn 和 speaker，其中 syn 是 SpeechSynthesizer 的实例对象；speaker 是 Talker 类的实例，然后在私有构造函数下对 syn 进行实例化。最后，通过 GetInstance 函数对外提供一个全局接口来访问 Talker 类实例。核心代码如下：

```
        using System.Speech.Synthesis;
    public   class Talker
{
    private SpeechSynthesizer syn;
    private static Talker speaker;
    private Talker()
    {
                syn = new SpeechSynthesizer();
    }

// <summary>
// 返回 Talker 实例
// </summary>
// <returns>Talker 对象</returns>
    public static Talker GetInstance()
    {
```

```
        if (speaker == null)
            speaker = new Talker();

        return speaker;
            }
```

3.2 功能实现

（1）使用 syn 对象朗读文本

```
    public void Speak(string text)
    {
        syn.SpeakAsync(text);
      }
```

（2）使用 syn 选择朗读人选项

```
    public void SelectVoice(string role)
    {
        syn .SelectVoice (role ); //role 是朗读人的姓名,可查
//看控制面板的语音属性中的语音选择下拉框
      }
```

（3）使用 syn 选择朗读音量（范围 0—100）

```
        public int Volume
    {
        get
                { return syn.Volume; }

        set
        {
           syn.Volume  =value ;
        }
      }
```

（4）使用 syn 合成 Wav 文件（语音导出功能）

```
    public void ExportToWave(string filename ,string text)
    {
        syn .SetOutputToWaveFile (filename );
        syn.Speak(text);
        syn.SetOutputToNull();
    }
```

（5）使用 syn 获取系统已安装的语音朗读角色

```
     // <summary>
    //获取系统安装的语音朗读角色
    // </summary>
    // <param name="culture">语音包语言,美式英文标识:
//0x409 或 en-US</param>
    // <returns>数组,数组保存了系统安装的语音朗读角
//色</returns>
    public string [] GetVoice(CultureInfo culture)
    {
        int i=0;

        string  [] voice=new string [syn .GetInstalledVoices
(culture ).Count] ;
        foreach  (var item in syn.GetInstalledVoices (culture
))

        {
                voice[i] = item.VoiceInfo.Name;
           i++;
        }

        return voice ;

    }
```

4 实现语音识别功能

对 SpeechRecognizer 实现二次封装，使用单例和观察者设计模式。当识别文字成功的时候，通过事件通知虚拟人来执行相应的动作。虚拟人通过 AxAgent 控件实现。具体做法分 3 步：

第一步：新建一个窗体取名 frmmain，右键单击工具箱窗口空白区，在弹出的下拉菜单中选择“选择项”菜单，在“选择工具箱项”的“COM 组件”选项卡中选择 Microsoft Agent Control 2.0，如图 2 所示，然后为 frmmain 窗体添加 Agent 控件。

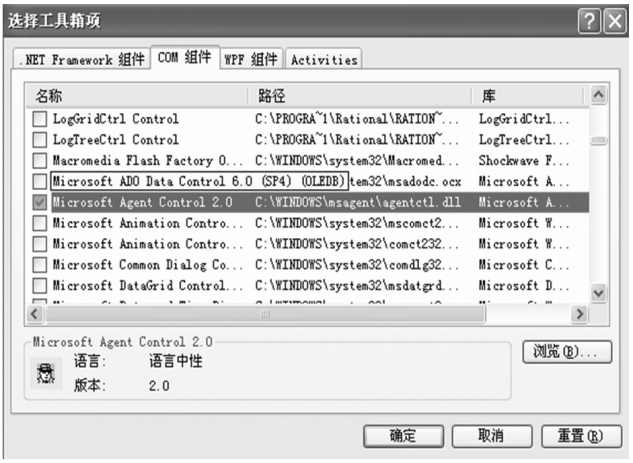

图 2

第二步：添加代码，初始化 Agent 控件对象 axAgent1。

```
                using AgentObjects;
                        using AxAgentObjects;
                 private void frmmain_Load (object sender,
EventArgs e)
               {
                axAgent1.BeginInit();
                 axAgent1.Characters.Load ("merlin", (object)@"
C:/Windows/Msagent/chars/merlin.acs");
//加载名为 merlin 虚拟人
```

```
                Character = axAgent1.Characters["merlin"];
                Character.LanguageID = 0x409;
//虚拟人支持的语言.0x409 为美式英语
                Character.Show(null);//显示 merlin 虚拟人
                }
```

第三步：自定义事件类 RecognizeEventArgs，**新建识别类** RecognitionEngine，**核心代码如下：**

```
        using System.Speech.Recognition;
        public class RecognizeEventArgs:EventArgs
    {
            public string message;
//message 保存了 axAgent1 控件对象要做的动作

            public RecognizeEventArgs(string message)
            {
                this .message =message ;
            }
    }
        public  class RecognitionEngine

{
        private SpeechRecognizer SRE;
            public event EventHandler<RecognizeEventArgs>
RecognitionEvent;//声明事件成员
            // <summary>
            // 识别
            // </summary>
            // <param name="text">识别的文字</param>
            // <param name="action">识别成功后的行为</
//param>
            public void Recognize(string text,string action)
            {  Grammar g;//声明语法对象
            GrammarBuilder gb;//声明语法规则构建器对象
                gb = new GrammarBuilder();
//初始化语法构建器对象
                gb.Culture = new CultureInfo(0x409);
//设置语法规则构件器实例的语言,本程序识别的是英文。
                gb.Append(text);//将识别内容添加到构建器中
            g = new Grammar(gb);
                g.SpeechRecognized += new EventHandler<
SpeechRecognizedEventArgs>(g_SpeechRecognized);
//注册识别事件
                SRE.LoadGrammar(g);//将语法装入识别器中。
//SRE 是 SpeechRecognizerde 的实例
                }
        // <summary>
            // 函数作用:识别到文字后,通知 frmmain 窗体上的
//axAgent1 控件对象,执行动作
            // </summary>
```

```
        private void g_SpeechRecognized (object sender,
SpeechRecognizedEventArgs e)
        {
            if (RecognitionEvent ! = null)
                RecognitionEvent (this, new Recog-
nizeEventArgs(action));
        }
```

在 Windows XP **下，可通过控制面板的语音选项中的“训练配置文件”提高识别率。操作方法如图** 3 **所示。**

图 3

5 结语

通过一个实例，介绍了语音合成与识别技术的实现，这种方法灵活性较大，适用性较强，所有源码均在 Windows XP+ C# 2008 **环境下调试成功。此范例可以扩展，如电子词典、英语口语训练。**

（**收稿日期**：2010-05-05）