Unity3d在Window上使用SAPI进行语音识别

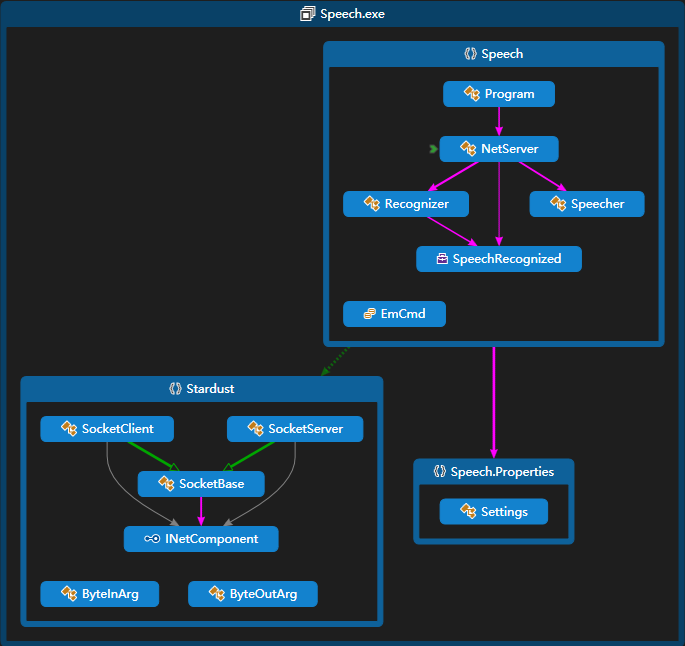
# 前言

在之前《[**Unity利用Sapi进行windows语音开发**](http://www.cnblogs.com/CodeGize/p/5419396.html)》中，本计划不准备继续做语音识别。因为在unity3d中已经提供了语音识别的相关方法，详见unity3d的官方文档：<https://docs.unity3d.com/ScriptReference/Windows.Speech.KeywordRecognizer.html>。但是有一点是这个支持win10。对于win7用户来说，如果不使用百度语音或者科大讯飞语音的话，那么使用SAPI就是最好的方式了。同样的，由于Unity中无法直接使用SAPI，所以只能按照原来的思路，把它写到一个exe工具中，然后再由unity3d来调用。

# 工程变更和重构

还是继续之前的工程，但是为了规范一些，我把工程名称从SpeechTest改为了Speech。并且重构了Socket通信结构，将原本的SocketExtra分为了SocketBase，SocketServer和SocketClient。这样做的目的就是简单化，并且增加SocketServer发送信息到SocketClient的功能。

这是工程的最终结构图



# 通信数据结构

由于speech加入了语音识别功能，所以为了区别之前的通信信息，重新整理定义了新的通信数据结构。通信数据结构定为二进制数据，前4位是一个int32型的命令码，这些命令码有不同的含义和内容：

* 1：表示初始化，
* 2：表示文字转语音，后面紧跟着一个string
* 3：表示语音识别，后面是一个int32型，0表示结束识别，1表示开始识别

当speech收到1时，进行初始化。收到2的信息时，读取之后的文本数据，然后交由Speecher来发音。收到3时，读取后面的数据，如果是1，则开始进行录音识别，如果是0则停止录音识别。

Speech代码NetServer.cs

public bool NetReciveMsg(byte[] recivebuffer, int netID)

{

var arg = new ByteOutArg(recivebuffer);

var cmd = arg.ReadInt32();

switch ((EmCmd)cmd)

{

case EmCmd.**Init**:

Init();

break;

case EmCmd.**Speak**:

var str = arg.ReadString();

Console.WriteLine(str);

m\_speecher.Speak(str);

return true;

case EmCmd.**Recognize**:

var scmd = arg.ReadInt32();

if (scmd == 1)

m\_recognizer.BeginRec();

if (scmd == 0)

m\_recognizer.EndRec();

return true;

default:

throw new ArgumentOutOfRangeException();

}

return false;

}

在unity发送的代码。（ByteInArg是一个简单地写如byte[]数据的类）

/\*\*\*测试代码，可删除Start\*\*\*/

public void OnGUI()

{

if (GUILayout.Button("Connect"))

{

StartCoroutine(Connect());

}

if (GUILayout.Button("InitServer"))

{

StartCoroutine(InitServer());

}

if (GUILayout.Button("Speak"))

{

Speak("hello world");

}

if (GUILayout.Button("Recognize Start"))

{

Recognize(true);

}

if (GUILayout.Button("Recognize End"))

{

Recognize(false);

}

}

/\*\*\*测试代码，可删除End\*\*\*/

private void Recognize(bool tf)

{

var arg = new ByteInArg();

arg.Write(3);

arg.Write(tf ? 1 : 0);

NetSendMsg(arg.GetBuffer());

}

public IEnumerator InitServer()

{

var arg = new ByteInArg();

arg.Write(1);

NetSendMsg(arg.GetBuffer());

yield return 1;

}

public void Speech(string str)

{

if (m\_socket.Connected)

{

var arg = new ByteInArg();

arg.Write(2);

arg.Write(str);

//var bytes = Encoding.Default.GetBytes(str);

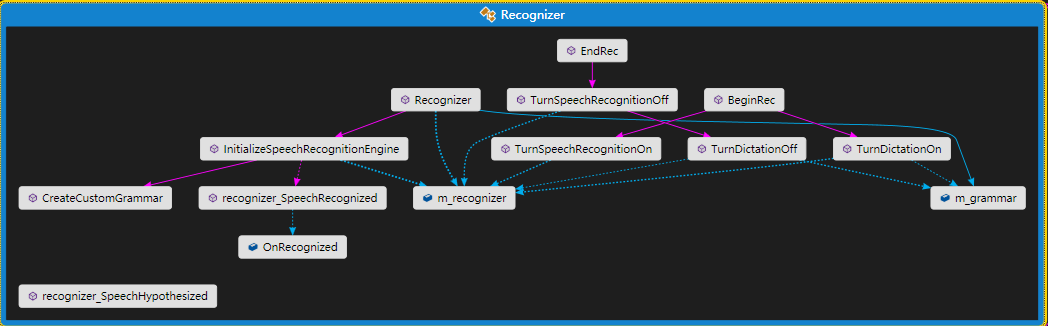
NetSendMsg(arg.GetBuffer());

}

}

# 封装SAPI语音识别模块

封装SAPI的语音识别模块这方面的代码很多，也比较简单。这里创建的Recognizer类，代码更改自<http://kevin19900306.iteye.com/blog/1206534>，其中的含义可以查阅SAPI的相关文档。Recognizer类有构造函数，两个接口，一个回调向外提供。这两个接口为开始识别BeginRec和停止识别EndRec，当识别完成时，使用回调，将识别的文字传出。



在speech的NetServer.cs中，回调相关的代码如下

public void Init()

{

m\_speecher = new Speecher();

m\_recognizer = new Recognizer

{

OnRecognized = OnRecognized

};

Console.WriteLine("初始化完成");

}

private void OnRecognized(string text)

{

var arg = new ByteInArg();

arg.Write(text);

NetSendMsg(arg.GetBuffer());

}

# 测试

运行unity3d，画面中出现5个按钮，分别为Connect，InitServer，Speak，Recognize Start，Recognize End。依次点击Connect，InitServer。然后点击Speak，听到“hello world”语音。点击Recognize Start，对麦克风说任何中文，可以看到debug输出对应的文字。点击Recognize End停止识别。

# Github源代码

源代码下载<https://github.com/CodeGize/UnitySapi/>

# 转载声明

转载请保留

http://[www.codegize.com](http://www.codegize.com/),

http://www.cnblogs.com/CodeGize/