

# 西华师范大学校园网视频下载工具的设计和实现

SinSoul

计算机科学与技术专业      指导教师:

**摘要:** 西华师范大学校园网视频下载工具是用于从西华师范大学的校园视频网抓取视频下载地址进行下载,同时能完全摆脱学校提供的视频插件限制,直接进行视频在线播放的第三工具。此工具极易使用,提供比校园网本身更好的搜索功能,自动化抓取所有剧集的下载地址,可自动调用迅雷等下载工具,并完成重命名工作。没有任何后台服务运行,不使用时不会占用系统资源。还提供自定义服务器地址功能,对其它使用相同服务器端的校园视频网具有通用性。

本论文主要介绍了校园视频网的工作原理分析,以及根据分析得到的校园网视频地址获取的方法,实现自动化获取下载地址,进行下载或者直接播放的过程。对各个功能模块的设计和实现过程都进行详细的说明。本工具在 Windows SDK 下使用 C++ 语言进行开发,使用基于 HTTP 协议的 C/S (客户端/服务器) 模式。此外对于 Windows 7 的 Aero 效果以及 GDI+ 在程序界面中的应用也有相应描述。

**关键词:** C++、HTTP、WinSocket、WinSDK、GDI+、Aero、视频下载

## China West Normal University Campus Network Video Hunter Design and Implementation

SinSoul

Computer Science and Technology      Instructor:

**Abstract:** The campus network video hunter is a link recovery tool that reveals the download link of the china west normal university campus network video site, meanwhile it can completely get rid of video plug-in, which is offered by campus, and directly play video online. It's easy to use this tool. The tool provides the better search functions than the campus itself does, grabs the download links about all of the episodes automatically. Call the download tools, such as Thunder, etc. and complete the renaming work as well. There is no background service operation, which means it will not take up system resources unless it is in service. Furthermore, it also provides custom server address, which has the generality towards other campus video network

with the condition of the same sever.

This paper introduces the campus network video site principle. As well as analyze the method of automated reveals the download link of videos for download or direct playback. Detailed description of the design and develop various functional modules. This tool used the Windows SDK with C++ language development, using the HTTP Client/Server mode. In addition, introduce how to enable Aero effects in the Windows 7 system and use GDI+ in user interface.

**Keywords:** C++,HTTP,WinSocket,WinSDK,GDI+,Aero,Video Download

## 目录

西华师范大学校园网视频下载工具的设计和实现.....	1
目录.....	2
第一章 引言 .....	4
1.1 工具开发的背景及必要性 .....	4
1.2 主要开发内容.....	4
第二章 开发环境及开发工具 .....	4
2.1 操作系统.....	4
2.2 编程语言及开发工具.....	5
2.2.1 C++语言.....	5
2.2.2 Microsoft Visual Studio 2010 10.0.40219.1 SP1Rel .....	5
2.2.3 Notepad++ 6.1.5.....	5
2.2.4 Google Chrome 23.0.1271.64 .....	5
2.3 其它辅助工具 .....	5
2.3.1 Visual Assist X 10.7.1903.0 built 2012.04.03 .....	5
2.3.2 VMware Workstation 8.0.1 build-528992.....	6
2.3.3 Nir Sofer SmartSniff v1.85.....	6
2.3.4 SVN 1.6.17 (r1128011) .....	6
第三章 校园视频网工作原理分析 .....	7
3.1 校园视频网简介及使用方法 .....	7
3.2 视频插件工作原理分析 .....	7
3.2.1 测试环境搭建.....	8
3.2.2 捕获数据分析.....	10
3.2.3 分析结果.....	10
第四章 校园网视频下载工具的设计.....	11
4.1 工具需求分析 .....	11
4.2 工具的概要设计.....	11
第五章 校园网视频下载工具的具体实现.....	12
5.1 HTTP 通信类的实现.....	12
5.1.1 HTTP 通信类概述.....	12

5.1.2 HttpMessenger 类关键成员函数及变量的介绍和使用 .....	12
5.1.3 HttpMessenger 使用示例 .....	13
5.2 界面实现 .....	14
5.2.1 界面组成介绍及实现 .....	14
5.2.2 Windows Aero 效果在程序中的使用 .....	17
5.2.3 GDI+在程序用户界面中的使用 .....	19
5.3 功能函数的实现 .....	21
5.3.1 更新和下载 Total.XML 文件 .....	21
5.3.2 搜索并在列表框显示符合要求的视频 .....	23
5.3.3 获取下载地址 .....	27
5.3.4 调用迅雷进行下载 .....	30
5.3.5 直接播放 .....	31
5.4 最终实现效果 .....	34
5.4.1 Aero 效果和“关于程序”界面的 GDI+图形绘制 .....	34
5.4.2 在 Windows XP 下初始下载 Total.xml 及“设置服务器地址”功能的演示 .....	35
5.4.3 从搜索到获取下载地址并下载播放器进行观看的演示 .....	36
5.4.4 批量下载及域名补丁效果 .....	36
第六章 测试 .....	37
6.1 西华师范大学本校内的测试 .....	37
6.2 其它院校的同学的测试 .....	37
第七章 总结 .....	37
7.1 关于开源 .....	37
7.1.1 开源的原因 .....	37
7.1.2 开源的方式 .....	37
7.2 总结 .....	38
参考文献: .....	38
致谢 .....	38

# 第一章 引言

## 1.1 工具开发的背景及必要性

西华师范大学在 2012 年 11 月底开通了校园高清视频网，以方便同学们进行观看并且减轻出口带宽压力。视频网是基于“光音网视高校云视频”服务搭建，提供丰富的正版影视内容，高速的局域网让观看视频不再需要等待缓冲。但是需要在安装学校提供的视频播放插件后才能访问校园视频网的资源，该插件的一个组件会注册成为系统服务一直运行，占用系统资源，而且只提供了在线播放功能，没有提供直接下载，这给想下载后放到电脑或手机上供以后观看的同学带来不便。在未安装该插件也是无法直接访问校园视频网页进行浏览的，必须通过工具访问。

为了解决上述问题，需要开发一个绿色小巧但功能强大的工具，以摆脱视频插件的束缚，让同学们更多的享受到校园视频网带来的好处。

## 1.2 主要开发内容

开发校园网视频下载工具的目的是提供一个绿色小巧易用的工具，能完全取代学校提供的视频插件，主要功能如下：

- (1) 提供方便快捷的途径抓取视频剧集的下载地址，进行下载或直接播放。
- (2) 提供比网站本身更强大的搜索功能
- (3) 实现在未安装插件时也能直接使用 IE 浏览器访问校园视频网页。
- (4) 提供自定义服务器地址的功能，让其它同样基于“光音网视高校云视频”服务搭建的校园视频网同样能使用本工具。

# 第二章 开发环境及开发工具

## 2.1 操作系统

由于此工具的受众群体为所有希望从学校视频网下载视频的同学，所以需要此工具能在绝大部份同学的计算机上运行，而目前同学们使用的主流操作系统是 Windows XP SP3 32 位以及 Windows 7 SP1 32 位和 Windows 7 SP1 64 位，于是将此工具的运行平台定为任意版本 Windows XP 和任意版本 Windows 7 甚至包括 Windows 8。

此外，学校所提供的视频插件只能运行于 Windows 平台，让使用 MAC OS X 或者是 Linux 的同学就完全不能访问校园视频网的资源，为了保证此工具能方便快捷的移植到类 Unix 系统，在开发过程中应该注意代码的可重用性，使之只需要做少许修改就能工作在类 Unix 平台上，让使用类 Unix 操作系统的同学同样能

方便的访问校园视频网的资源，更能体现此工具存在的价值。

## **2.2 编程语言及开发工具**

### **2.2.1 C++语言**

考虑到工具需要在 Windows XP 和 Windows7 都能正常工作，并且不是所有同学的计算机上都安装了 .NetFramework 平台或者是 Java 运行时环境，所以直接排除使用 C#和 Java 等语言。以及之后可能移植到 MAC 或者 Linux 系统，还有程序的可扩展和灵活性，决定使用 C++语言进行开发，并且尽量使用标准模板库（STL），而不是只有某个平台才支持的框架，比如 MFC。在 Windows 上也直接使用 Windows 提供的 API 进行开发，这样在移植到其它系统时只需要对不同的 API 进行修改即可。

### **2.2.2 Microsoft Visual Studio 2010 10.0.40219.1 SP1Rel**

本论文主要讨论在 Windows 系统下的开发过程，使用的集成开发环境（IDE）为 Visual Studio 2010。

Visual Studio 包含基于组件的开发工具（如 Visual C#、Visual J#、Visual Basic 和 Visual C++），以及许多用于简化基于小组的解决方案的设计、开发和部署的其他技术。

Visual Studio 2010 集成开发环境（IDE）的界面被重新设计和组织，变得更加清晰和简单。新的 IDE 更好的支持了多文档窗口以及浮动工具窗，并且对于多显示器的支持也有所增强。IDE 的外壳使用 WPF 重写，内部使用 MEF 重新设计，以提供比先前版本更好的扩展性。

### **2.2.3 Notepad++ 6.1.5**

除了 VS2010 外，开发过程中还使用 Notepad++做为代码和文件编辑器。

Notepad++是一套自由软件的纯文本编辑器，由台湾人侯今吾基于同是开放源代码的 Scintilla 文本编辑组件并独力研发。该软件以 GPL 发布，有完整的中文化接口及支持多国语言撰写的功能（采用万国码 UTF-8 技术）。不仅有语法高亮度显示，也有语法折叠功能，并且支持宏以及扩充基本功能的外挂模块。由于在资源消耗方面与微软的记事本大体相当，但功能更为强大。除了可以用来制作一般的纯文字的帮组文档，也十分适合用作撰写电脑程序的编辑器。

### **2.2.4 Google Chrome 23.0.1271.64**

在对校园视频网进行测试的过程中，使用 Chrome 浏览器来发送请求，并观察服务器响应，其提供的网页调试功能能很方便地观察 HTTP 数据的请求和响应，查看网页源代码，调试 Javascript 代码等。

## **2.3 其它辅助工具**

### **2.3.1 Visual Assist X 10.7.1903.0 built 2012.04.03**

为了加速应用程序的开发，开发过程中使用 VS 插件 VAX，它提供自动识别各种关键字，系统函数，成员变量，自动给出输入提示，自动更正大小写错误，自动标示错误等各种实用的功能。

### **2.3.2 VMware Workstation 8.0.1 build-528992**

为了方便对校园网视频插件进行分析，和对编写的校园网视频下载工具进行测试，使用安装了 Windows XP SP3 的 VMware 虚拟机以辅助开发。

### **2.3.3 Nir Sofer SmartSniff v1.85**

对校园视频网进行分析的过程中需要对视频插件所发送和接收的数据包进行分析，就需要使用具有抓取数据包功能的软件进行辅助，这里并没有采用 WireShark 这样功能强大但复杂的软件，而使用小巧的 SmartSniff 进行数据包的捕获。

SmartSniff 是一款 TCP/IP 数据包捕获软件，允许你检查经过你的网络适配器的网络传输。该软件的双层界面显示了捕获的数据包和在 ASCII 或者十六进制格式下的详细的信息。额外的功能包括本地和远程传输的代码进行着色。

### **2.3.4 SVN 1.6.17 (r1128011)**

校园网视频下载工具会是一个开源项目，使用的版本控制软件为 Subversion，简称 SVN，是一个开放源代码的版本控制系统，相对于的 RCS、CVS，采用了分支管理系统，它的设计目标就是取代 CVS。

## 第三章 校园视频网工作原理分析

### 3.1 校园视频网简介及使用方法

西华师范大学的校园视频网站需要进入学校主页->公共服务->电子资源，下载并安装客户端软件后才能使用。

在安装完成后，是名为“校园高清视频”的软件，打开软件后界面如图 3.1 所示，可以选择或者搜索自己喜爱的影片，进入影片描述页面，点击播放按钮进行观看。



图 3.1 校园视频网客户端展示

### 3.2 视频插件工作原理分析

在整个网站和软件中并没有直接提供视频下载，想要获取到下载地址，因此需要分析该客户端是如何获取视频地址，并进行播放的。

从客户端软件界面来看，这是一个内嵌浏览器控件的软件，拥有地址栏，标签栏等普通浏览器的特点，其播放界面如图 3.2 所示，同样是一个类浏览器的界面。

从这些地方还不能看出视频插件是如何获取到下载地址的，需要进一步捕获客户端与服务器通信的数据加以分析。



图 3.2 校园视频网客户端播放器界面展示

### 3.2.1 测试环境搭建

为了不受到其它软件所发送数据包的影响，尽量只抓取视频插件所发送和接收的数据，将视频插件安装在系统为干净的 Windows XP 的虚拟机当中，视频插件发送和接收的所有数据都通过通过虚拟机的网卡，该虚拟机使用 NAT 方式访问网络，数据会通过物理机上的“VMware Network Adapter VMnet8”虚拟网卡，此时只需要在物理机上使用数据包捕获软件捕获这张网卡上所有的数据就可以了。

在 SmartSniff 和“Options”->“Capture Options”中设置捕获方法(Capture Method)为原始套接字(Raw Sockets)，网卡选择 VMnet8 虚拟网卡，如图 3.3 所示。

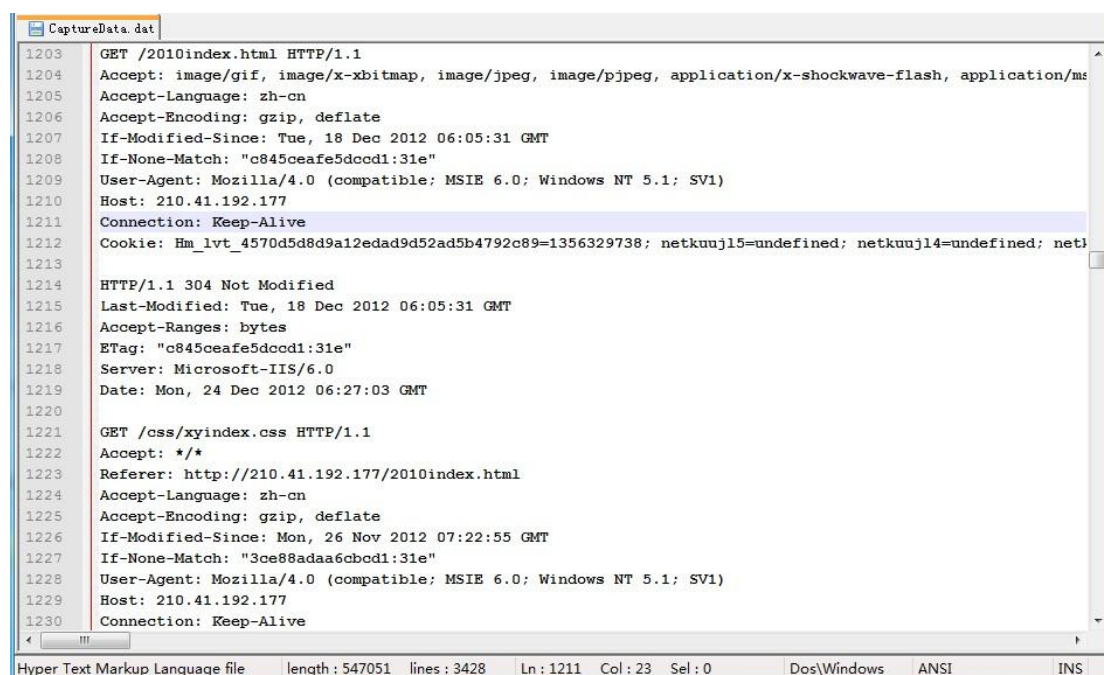




### 3.2.2 捕获数据分析

捕获到的数据中大多为 HTTP 协议的数据包，有少量服务名称为 bootpc、netbios-ns、netbios-dgm 的数据包，这是由视频插件所带的“宝丽通 PXP 流媒体加速”工具所产生的，对分析没有多大作用，所以忽略。

将捕获的数据导出为 dat 文件，部分数据如图 3.5 所示，完整数据请见附件中的 CaptureData.dat 文件。



```
1203 GET /2010index.html HTTP/1.1
1204 Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/ms
1205 Accept-Language: zh-cn
1206 Accept-Encoding: gzip, deflate
1207 If-Modified-Since: Tue, 18 Dec 2012 06:05:31 GMT
1208 If-None-Match: "c845ceafe5dcd1:31e"
1209 User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
1210 Host: 210.41.192.177
1211 Connection: Keep-Alive
1212 Cookie: Hm_lvt_4570d5d8d9a12edad9d52ad5b4792c89=1356329738; netkuuj15=undefined; netkuuj14=undefined; netl
1213
1214 HTTP/1.1 304 Not Modified
1215 Last-Modified: Tue, 18 Dec 2012 06:05:31 GMT
1216 Accept-Ranges: bytes
1217 ETag: "c845ceafe5dcd1:31e"
1218 Server: Microsoft-IIS/6.0
1219 Date: Mon, 24 Dec 2012 06:27:03 GMT
1220
1221 GET /css/xyindex.css HTTP/1.1
1222 Accept: */*
1223 Referer: http://210.41.192.177/2010index.html
1224 Accept-Language: zh-cn
1225 Accept-Encoding: gzip, deflate
1226 If-Modified-Since: Mon, 26 Nov 2012 07:22:55 GMT
1227 If-None-Match: "3ce88adaa6cbd1:31e"
1228 User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
1229 Host: 210.41.192.177
1230 Connection: Keep-Alive
```

图 3.5 部分截取数据包的内容

将捕获数据中的认为有用的 HTTP 请求还原为 HTTP 链接，并尝试更改链接中的参数，使用 Chrome 浏览器再次发出，并在 Chrome 的网页调试器中观察响应结果，以获取更详细过程。

### 3.2.3 分析结果

通过不断的尝试，大致得到了视频插件获取视频地址的方法。

```
获取影片列表，这个文件保存了服务器上所有电影名称和ID，可下载到本地使用
http://210.41.192.177/mov/xml/Total.xml

获取下载地址，####为集数，从0开始计数，如果是只有一集的电影，则####值为0。
$$$部分为上面那个文件中保存的电影ID。打开这个网址就能得到下载链接
http://210.41.192.177/xy_new.asp?a=####&b=$$$&time=2012-11-2822:35:41

获取影片描述，同上。
http://210.41.192.177/mov/$$$$/film.xml

获取影片分集名称？可能是这样，基本无用。
http://210.41.192.177/mov/$$$$/url.xml

搜索页面，比如搜索：想你
http://210.41.192.177/mxzcassjg.html?key=想你

为了能在不安装那破客户端软件情况下正常访问网页，还需要添加如下hosts记录
#校园电影网
210.41.192.177 mov.csonline.com.cn
```

图 3.6 获取影片下载地址的方法

## 第四章 校园网视频下载工具的设计

### 4.1 工具需求分析

通过前面的分析，想要实现自动化的获取下载地址，实际上就是需要模拟浏览器的行为，发送 HTTP 请求，并根据服务器响应来做出下一步判断，最终得到所有下载地址。

具体包括：

- (1) 得到某部电影的 ID，可以由直接用户输入，可以根据用户输入的关键字进行确定，也可以从链接地址中获取。
- (2) 根据 ID 获取影片描述。
- (3) 根据 ID 尝试获取该视频的所有下载地址。
- (4) 根据用户操作自动调用迅雷进行下载，或者调用播放器直接进行播放。
- (5) 提供让用户自定义服务器地址的接口。
- (6) 根据用户定义的服务器地址，修改 HOSTS 文件，以达到使用浏览器就能直接访问校园视频网的效果。

### 4.2 工具的概要设计

校园网视频下载工具的主要功能是抓取视频的下载地址，其抓取过程的流程图，如图 4.1 所示。

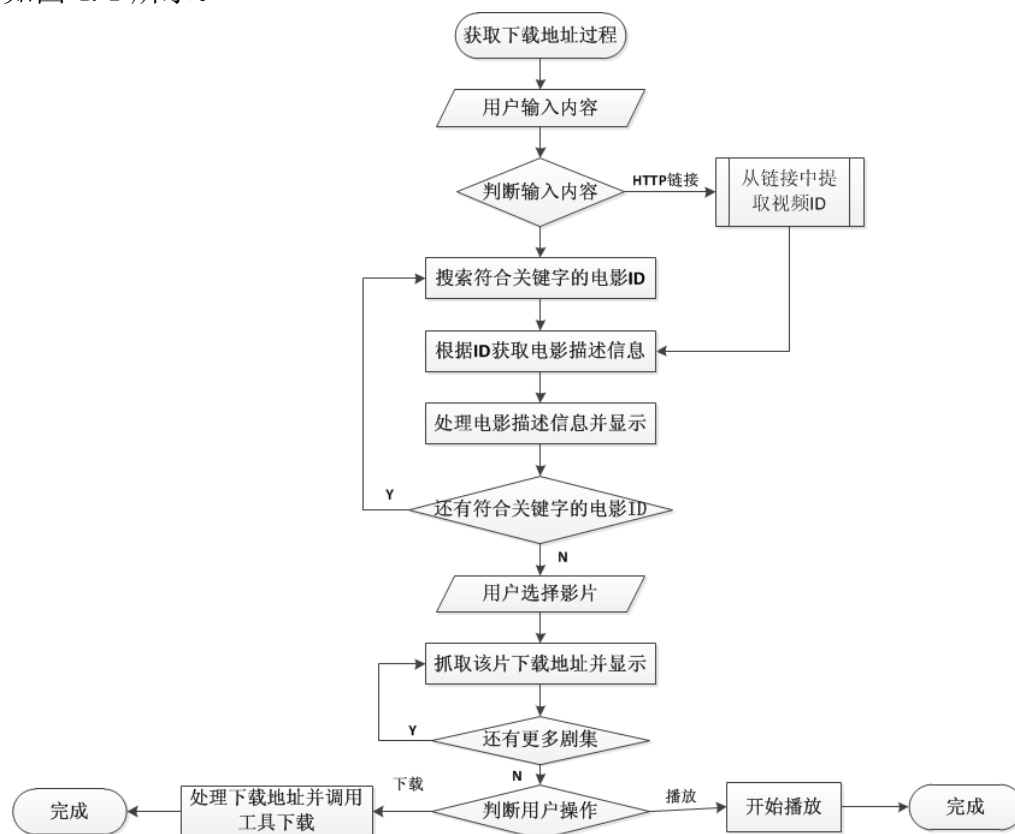


图 4.1 获取下载地址流程

## 第五章 校园网视频下载工具的具体实现

### 5.1 HTTP 通信类的实现

#### 5.1.1 HTTP 通信类概述

校园网视频下载地址的获取实际上是很多次使用 HTTP 协议通信，来获取数据的过程，那么实现一个稳定强壮的 HTTP 通信类就非常有必要了，之所以不使用诸如 WinInet 系列的函数，是因为这些函数是 Windows 独有，为了方便程序能很快的移植到类 Unix 平台，同时也为了程序拥有更强的控制能力，这里基于基本的 socket 编程，实现了一个 HTTP 通信类，名称为 HttpMessenger。

经过作者长时间的测试和修改，已经能同诸如 IIS, nginx, Apache 等流行的 HTTP 服务软件进行稳定的通信，并且能轻松改写用于类 Unix 平台。

#### 5.1.2 HttpMessenger 类关键成员函数及变量的介绍和使用

(1) 构造函数: `HttpMessenger(char *szHostName="127.0.0.1", int nPort=80);`

这是HttpMessenger类的构造函数，共两个参数，第一个为要访问的服务器的地址，第二个参数为通过哪个端口进行访问，由于是用于HTTP通信，默认值为80端口。

(2) 错误查询函数: `string HttpMessenger::GetErrorLog();`

此函数用于在某个步骤失败时查询具体的出错原因和出错位置，以便于及时发现错误并修正，或用于显示提示。

(3) 创建并发送HTTP请求: `bool CreateAndSendRequest(char *RequestType, char *ResourcePath, char *Host=NULL, char *PostData=NULL, bool OnlyGetLength=false, char *Save2File=NULL);`

此函数根据输入的参数创建HTTP请求头，并自动向服务器发送请求和接收服务器的响应，共有6个参数。

RequestType为向服务器发送的HTTP请求的类型，通常有GET和POST等。

ResourcePath为请求的资源的路径。

Host为提交请求的服务器地址。

PostData, 这个参数在使用POST类型的请求时使用，是向服务器发送的数据，如果是GET请求，请将此参数传为NULL。

OnlyGetLength, 这个参数为true时，仅仅只获取到服务器返回的资源长度后，函数就马上返回，并不会接收服务器传回的数据。

Save2File, 此参数不为NULL时，会把传入的字符串当做一个文件名，或者是文件路径，来创建一个文件，在接收数据的过程中，把接收到的数据

写入文件当中，可以用于较大文件下载。

如果请求成功则返回 true，连接失败则返回 false，此时可以调用 `GetErrorLog()` 函数获取具体失败的原因。

(4) 获取HTTP响应的状态: `int GetResponseState();`

此函数用于查看服务器返回的HTTP响应状态，常见的返回值有200：请求成功，302：资源被移动，404：请求的资源未找到等等，更多内容请参考HTTP协议标准。

(5) 初始化套接字函数: `static bool InitialSocket();`

此静态方法需要在程序的初始的地方调用一次，用于初始化套字，否则类不能正常工作。

(6) HTTP请求头的内容: `string m_RequestHeader;`

访问此成员变量可以得到所生成的HTTP请求头。

(7) HTTP响应头的内容: `string m_ResponseHeader;`

访问此成员变量可以得到服务器返回的HTTP响应头。

(8) HTTP 响应的全部内容: `string m_ResponseText;`

访问此成员就是可以得到服务器返回的 HTTP 响应内容，如果已经指定在函数 `CreateAndSendRequest` 中将请求内容保存为文件，此成员内容为空，请读取文件进行获取。

(9) 向服务器发起连接请求: `bool CreateConnection();`

此函数用于向服务器发起连接请求，如果连接成功则返回 true，连接失败则返回 false，此时可以调用 `GetErrorLog()` 函数获取具体失败的原因。

### 5.1.3 HttpMessenger 使用示例

(1) 在程序开始处，使用 `HttpMessenger::InitialSocket();` 加载套接字，此操作只需要做一次。

(2) 创建 `HttpMessenger` 对象，例如：

```
HttpMessenger *hm_GetTotalXML=new HttpMessenger(server_addr);
```

(3) 向服务器发起连接，例如：

```
if (!hm_GetTotalXML->CreateConnection())
{
    Msg("检测更新时，连接服务器失败，错误日志:%s\r\n",
        (char *)hm_GetTotalXML->GetErrorLog().c_str());
    delete[] hm_GetTotalXML;
    return;
}
```

(4) 连接成功后，创建 HTTP 请求头，并向服务器发送请求，例如：

```
if (!hm_GetTotalXML->CreateAndSendRequest("GET",
    "/mov/xml/Total.xml",server_addr,NULL,true))
```

```

{
    Msg("检测更新时，发送请求失败，错误日志:\r\n%s",
        hm_GetTotalXML->GetErrorLog().c_str());
    delete[] hm_GetTotalXML;
    return;
}

```

(5) 整个请求过程完成，根据需要来处理得到的响应内容。

## 5.2 界面实现

校园网视频下载工具的界面没有使用任何第三方界面库或者框架，直接使用 Windows SDK 的 API 来创建。

### 5.2.1 界面组成介绍及实现

为了极大的简化用户的操作，提升用户体验，工具使用极简的界面设计风格，仅一个文本框供用户输入，其余两个文本框用于进度显示和提示信息显示，一个多功能按钮和一个显示影片信息的列表框，以及菜单栏。其创建过程包括：

(1) 创建主窗口，主窗口是一个 720\*520 大小的基本窗口，创建代码如下：

```

int WINAPI WinMain (HINSTANCE hThisInstance,HINSTANCE
hPrevInstance,LPSTR lpszArgument,int nFunsterStil)
{
    MSG messages;
    WNDCLASSEX wincl;
    hinst=hThisInstance;
    wincl.hInstance = hThisInstance;
    wincl.lpszClassName = szClassName;
    wincl.lpfnWndProc = WindowProcedure;
    wincl.style = CS_DBLCLKS;
    wincl.cbSize = sizeof (WNDCLASSEX);
    wincl.hIcon = LoadIcon (NULL,MAKEINTRESOURCE (IDI_ICON1));
    wincl.hIconSm =
(HICON)LoadImage(hThisInstance,MAKEINTRESOURCE (IDI_ICON1),IMAGE_ICON,

    GetSystemMetrics (SM_CXSMICON),GetSystemMetrics (SM_CYSMICON),LR_DE
FAULTCOLOR);
    wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
    wincl.lpszMenuName = MAKEINTRESOURCE (IDR_MENU1);
    wincl.cbClsExtra = 0;
    wincl.cbWndExtra = 0;
    if (!RegisterClassEx (&wincl))
    {
        return 0;
    }
    hwnd = CreateWindowEx (NULL,szClassName,wndName,
    WS_OVERLAPPED|WS_SYSMENU|WS_MINIMIZEBOX,CW_USEDEFAULT,CW_USEDEFAU
LT,720,520,HWND_DESKTOP,NULL,hThisInstance,NULL);

```



```

ShowWindow (hwnd, nFunsterStil);
while (GetMessage (&messages, NULL, 0, 0))
{
    TranslateMessage (&messages);
    DispatchMessage (&messages);
}
Gdiplus::GdiplusShutdown (gdiplusToken);
DeleteFile ("VideoList.m3u");
return messages.wParam;
}

```

(2) 按钮以及编辑框的创建方法相同，代码如下：

```

hwndButton=CreateWindowEx (NULL,TEXT ("BUTTON"),TEXT ("捕获"),
    WS_CHILD|WS_VISIBLE|BS_DEFPUSHBUTTON|BS_PUSHBUTTON
    ,370,28,80,25,hwnd, (struct HMENU__ *)1,hThisInstance,NULL);
hwndEdit=CreateWindowEx (NULL,TEXT ("EDIT"),TEXT (""),
    WS_CHILD|WS_VISIBLE|ES_AUTOHSCROLL,
    60,30,300,20,hwnd, (struct HMENU__ *)0,hThisInstance,NULL);
hwndLable_0=CreateWindowEx (NULL,TEXT ("EDIT"),TEXT (""),
    WS_CHILD|WS_VISIBLE|WS_VSCROLL|ES_MULTILINE|ES_READONLY,
    15,250,690,200,hwnd, (struct HMENU__ *)0,hThisInstance,NULL);
hwndLable_1=CreateWindowEx (NULL,TEXT ("EDIT"),TEXT (""),
    WS_CHILD|WS_VISIBLE|ES_READONLY|ES_MULTILINE,
    460,30,200,20,hwnd, (struct HMENU__ *)0,hThisInstance,NULL);

```

(3) 列表控件的创建：

在 Windows 中，列表控件为通用控件中的一个，在使用前需要使用函数来初始化和加载控件相关资源，函数名称 InitCommonControls(); 并且需要链接库文件 Comctl32.lib。

列表控件的创建和初始化函数如下：

```

HWND CreateListView (HWND hwndParent, LPSTR szWindowName)
{
    HWND hWndListView;
    hWndListView =
CreateWindowEx (WS_EX_CLIENTEDGE, "SysListView32", NULL, LVS_REPORT|WS_CH
ILD|WS_VISIBLE| LVS_EDITLABELS, CW_USEDEFAULT, CW_USEDEFAULT,
CW_USEDEFAULT, CW_USEDEFAULT, hwnd, (struct HMENU__ *)0, hinst, NULL);
    if (hWndListView == NULL)
    {
        Msg ("创建列表控制失败\r\n");
        return NULL;
    }
    if (InitListViewColumns (hWndListView) )
    {

```

```

        return hWndListView;
    }
    return false;
}
bool InitListViewColumns(HWND hWndListView)
{
    char szText[256];
    LVCOLUMN lvc;
    LPSTR ColNames[] = {"视频名称", "导演", "主演", "频道", "ID"};
    int Collong[]={200,100,200,100,86};
    lvc.mask = LVCF_FMT | LVCF_WIDTH | LVCF_TEXT | LVCF_SUBITEM;
    lvc.pszText = szText;
    lvc.iImage = 1;
    lvc.fmt = LVCFMT_LEFT;
    for (int index=0;index<5;index++)
    {
        lvc.cx =Collong[index];
        lvc.pszText = ColNames[index];
        lvc.iSubItem = 0;
        if (ListView_InsertColumn(hWndListView, index, &lvc) == -1)
        {
            return false;
        }
    }
    return true;
}

```

#### (4) 菜单栏的创建

本程序中菜单栏使用 Visual Studio 2010 创建菜单资源文件。共包括两个菜单项，分别是“工具”和“帮助”。工具菜单项主要是校园网视频下载工具所提供的各种功能，包括：

- a. 使用迅雷下载:此功能在用户抓取完影片下载地址后使用,用于调用迅雷提供的 Com 接口进行文件的下载,视频下载工具应对抓取到的下载链接进行处理,生成合适的文件名等。
- b. 生成列表并播放:此功能同样在用户抓取影片下载地址后使用,用于生成 m3u 格式的播放列表,并且调用 Mplayer 进行播放。
- c. 检查列表更新:此功能应该在程序初始化时自动调用一次,以检查本地的 Total.xml 文件(此文件的描述见 3.2.3 节)是否与服务器一致,以确保用户能搜索到最新更新的电影,并且也可以让用户在自行设定服务器后手动调用。
- d. 设置服务器地址:此功能提供一个用户能自定义服务器地址的接口,让非西华师范大学,但其学校同样也是使用“光音网视高校云视频”服务搭建视频网的同学同样能使用本工具进行视频下载。



- e. 域名补丁：此功能用于修改系统的 HOSTS 文件，以保证校园视频网所使用的域名能正确的解析到本学校的视频服务器地址，实现使用普通的 IE 浏览器也能直接访问校园视频网，而不必通过视频插件才能访问。
- f. 访问网页：使用 IE 浏览器访问校园视频网主页。
- g. 退出程序：退出程序并完成临时文件清理工作。

帮助菜单项中主要有三个选项，分别为：

- a. 关于作者：显示作者信息，以便使用者在遇到软件问题时能联系到作者。
- b. 关于程序：显示程序简介信息和一些使用技巧、注意事项等。
- c. 项目地址：本程序会是一个开源项目，此功能用于访问项目主页。
- d. 检查更新：用于检查本程序是否有更新的版本，并完成更新工作。

### 5.2.2 Windows Aero 效果在程序中的使用

由于本工具界面非常简单，为了界面的美观，决定在支持 Windows Aero 效果的系统上启用 Aero 效果，以美化程序界面。

Windows Aero 是从 Windows Vista 开始使用的新型用户界面，透明玻璃感让用户一眼贯穿。“Aero”为四个英文单字的首字母缩略字：Authentic（真实）、Energetic（动感）、Reflective（反射）及 Open（开阔）。意为 Aero 界面是具立体感、令人震撼、具透视感和阔大的用户界面。除了透明的接口外，Windows Aero 也包含了实时缩略图、实时动画等窗口特效，吸引用户的目光。

前面已经说过，校园网视频下载工具需要运行的平台包括了任意版本的 Windows XP，而 Windows XP 和部分基本版的 Windows 7 并不支持 Aero 效果，如果在程序中直接启用，会让程序无法在 XP 中运行，所以程序需要自行判断当前系统是否支持 Windows Aero 效果，然后再决定是否启用。

具体的实现代码如下：

```
bool EnableAero(HWND hwnd_aero)
{
    typedef struct _MARGINS
    {
        int cxLeftWidth;        // width of left border that retains its size
        int cxRightWidth;       // width of right border that retains its size
        int cyTopHeight;        // height of top border that retains its size
        int cyBottomHeight;     // height of bottom border that retains its size
    } MARGINS, *PMARGINS;
    HMODULE library=LoadLibrary("dwmapi.dll");
    if(0!= library)
    {
        HRESULT (WINAPI * DICE) (BOOL *pfEnabled);
        HRESULT (WINAPI * DEFICA) (HWND hw,MARGINS *pMarInset);
```

```

        DICE=(HRESULT (WINAPI*) (BOOL
*))GetProcAddress(library,"DwmIsCompositionEnabled");
        DEFICA=(HRESULT (WINAPI*) (HWND,MARGINS
*))GetProcAddress(library,"DwmExtendFrameIntoClientArea");
        if (DICE!=NULL&&DEFICA!=NULL)
        {
            BOOL bDwm;
            DICE(&bDwm);
            if (bDwm&&hwnd_aero==NULL)
            {
                return true;
            }
            SetWindowLong(hwnd_aero,GWL_EXSTYLE,GetWindowLong(hwnd_aero,GWL_EXSTYLE)|0x80000);
            SetLayeredWindowAttributes(hwnd_aero,TRANSPARENTCOLOR, 0, LWA_COLORKEY);
            if(bDwm)
            {
                MARGINS mrg = {-1};
                __asm
                {
                    lea eax,mrg
                    push eax
                    push hwnd_aero
                    call DEFICA
                }//DEFICA(hwnd_aero,&mrg); //妈的直接调用要报错，汇编一切正常
                return true;
            }
            FreeLibrary(library);
            return false;
        }
    }
    return false;
}

```

需要注意的是，TRANSPARENTCOLOR 宏为指定哪个颜色做为透明色，最好使用界面上不太常出现的颜色，如果使用的是 RGB 的黑色（0x00000000）与 ARGB 的 100%透明的二进制表示完全一样，再加上文本框中的文字，按钮上的文字等同样使用黑色画刷来绘出文本，所以 DWM 将以为这部分内容也将呈现为半透明效果，会出现比较难看的效果。一个解决方案就是手工重新绘制这些文本框和按钮。而不应把时间花费在“手工绘制”每一个控件上。在这里使用系统已经自带的那一套控件，而不是去重复发明轮子。

更实际的做法就是使用分层窗体（layered window）。分层窗体最先由 Windows 2000 引入，它支持 Alpha 混合，自然成了实现玻璃效果的理想图景。

分层窗体提供了两种截然不同的编程模型。你既可以使用 UpdateLayeredWindow 函数提供一个与设备无关的位图(不方便),完整定义屏幕上窗体的整体样式,也可以直接使用 SetLayeredWindowAttributes 函数,本论文在实现的过程中使用的是后者。

### 5.2.3 GDI+在程序用户界面中的使用

为了更美观的显示一些提示信息,或者是在程序界面上显示一些图像,比如显示出视频截图或是电影封面等影片相关的简介信息。虽然目前的设计中并不包含这部分功能,但从长远来看很有必要留出,以便于将来有更多开发者想加入此功能时能快速方便的实现。对于这些信息在界面上的绘制,本程序中使用的是 GDI+。

GDI+从 Windows XP 操作系统开始引入,提供二维的矢量图形,改进旧有的 GDI,加强的可视化属性,例如边界,渐变和透明。通过 GDI+,能够直接将 BMP 转成 JPG 或其它格式的图片,还能够生成 SVG、Flash 等。GDI+ 使用 ARGB 的值来表示颜色。GDI+的双缓冲技术可以提高绘图效率,可避免屏幕闪烁。

在使用 GDI+前需要初始化 GDI+,即调用函数:

```
Gdiplus::GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL);
```

在程序退出前需要释放 GDI+,调用函数:

```
Gdiplus::GdiplusShutdown(gdiplusToken);
```

我在程序中实现了三个相关函数:

- a. ImageFromIDResource:从程序的资源文件中加载图像文件,用于绘制。
- b. GDIPlusDrawImage: 将图像文件绘制在界面的指定位置上。
- c. GDIPlusDrawText: 将文字绘制在界面的指定位置上。

具体的实现代码如下:

```
bool ImageFromIDResource(UINT nID, LPCTSTR sTR, IStream* &pstm)
{
    HRSRC hRsrc = FindResource(hinst, MAKEINTRESOURCE(nID), sTR);
    if (!hRsrc)
    {
        return false;
    }
    DWORD len = SizeofResource(hinst, hRsrc);
    BYTE* lpRsrc = (BYTE*)LoadResource(hinst, hRsrc);
    if (!lpRsrc)
    {
        return false;
    }
    HGLOBAL m_hMem = GlobalAlloc(GMEM_FIXED, len);
    BYTE* pmem = (BYTE*)GlobalLock(m_hMem);
```

```

        memcpy(pmem,lpRsrc,len);
        CreateStreamOnHGlobal(m_hMem,false,&pstm);
        GlobalUnlock(m_hMem);
        FreeResource(lpRsrc);
        return true;
    }

void GDIPlusDrawImage(HDC hdc,UINT SourceID,LPCTSTR SourceDIR,
    Gdiplus::REAL leftx,Gdiplus::REAL lefty,
    Gdiplus::REAL rightx,Gdiplus::REAL righty)
{
    using namespace Gdiplus;
    Graphics graphics(hdc);
    IStream* pstm=NULL;
    ImageFromIDResource(SourceID,SourceDIR,pstm);
    Bitmap bitmap(pstm);
    graphics.SetInterpolationMode(InterpolationModeHighQualityBicubic);
    graphics.DrawImage(&bitmap,leftx,lefty,rightx,righty);
}

void GDIPlusDrawText(HDC hdc,PWCHAR string,Gdiplus::REAL x_point=0,
    Gdiplus::REAL y_point=0,PWCHAR userFont=L"arial",Gdiplus::REAL
    fontSize=12,Gdiplus::Color pColor=Gdiplus::Color(0,0,0))
{
    using namespace Gdiplus;
    Graphics graphics(hdc);
    SolidBrush brush(pColor);
    PointF pointF(x_point, y_point);
    graphics.SetTextRenderingHint(TextRenderingHintAntiAliasGridFit);
    FontFamily fontFamily(userFont);
    if (fontFamily.IsAvailable())
    {
        Gdiplus::Font font(&fontFamily,fontSize, FontStyleRegular,
UnitPixel);
        graphics.DrawString(string,
(INT)wcslen(string),&font,pointF,&brush);
    }
    else
    {
        FontFamily defaultFont(L"arial");
        Gdiplus::Font font(&defaultFont,fontSize,FontStyleRegular,
UnitPixel);
        graphics.DrawString(string,
(INT)wcslen(string),&font,pointF,&brush);
    }
}

```

```
}
```

### 5.3 功能函数的实现

本节主要介绍视频下载工具中基本功能函数的作用和实现方法。

#### 5.3.1 更新和下载 Total.XML 文件

Total.xml 文件中已经包括了服务器上所有视频文件的基本信息，包括视频 ID，视频名称等，视频下载工具的搜索功能实际上就是对 Total.xml 文件进行搜索，所以从服务器下载 Total.xml 以及检测 Total.xml 的更新是首先要实现的功能。对于检测更新，实现中使用文件大小比较的方法，先获取本地文件的大小，然后再向服务器发送仅获取文件大小的 HTTP 请求，两者进行比较，如果不同，则说明服务器上的 Total.xml 已经更新，程序需要重新从服务器下载。

具体的下载和更新函数实现如下：

```
bool DownloadXML(string *TotalXML)
{
    HANDLE hFile=NULL;
    DWORD dwWrite=0;
    HttpMessenger *hm_GetTotalXML=new HttpMessenger(server_addr);
    if (!hm_GetTotalXML->CreateConnection())
    {
        Msg("下载列表时，连接服务器失败，错误日志:%s\r\n", (char
*)hm_GetTotalXML->GetErrorLog().c_str());
        delete[] hm_GetTotalXML;
        return false;
    }
    if (!hm_GetTotalXML->CreateAndSendRequest("GET",
        "/mov/xml/Total.xml",server_addr)){
        Msg("下载列表时，发送请求失败，错误日志:\r\n%s",
        hm_GetTotalXML->GetErrorLog().c_str());
        delete[] hm_GetTotalXML;
        return false;
    }
    hFile=CreateFile("Total.xml",GENERIC_WRITE,0,0,CREATE_ALWAYS,FILE
_ATTRIBUTE_NORMAL,0);
    WriteFile(hFile, (char *)hm_GetTotalXML->m_ResponseText.c_str(),
        hm_GetTotalXML->m_ContentLength,&dwWrite,NULL);
    TotalXML->erase();
    *TotalXML+=hm_GetTotalXML->m_ResponseText.c_str();
    delete [] hm_GetTotalXML;
    CloseHandle(hFile);
    return true;
}

void LoadTotalXML(void *TotalXML)
{

```

```

    EnableWindow(hWndButton, false);
    DWORD dwRead, dwDataSize;
    HANDLE
hFile=CreateFile("Total.xml", GENERIC_READ, 0, 0, OPEN_EXISTING, FILE_ATTR
IBUTE_NORMAL, 0);
    if (hFile==INVALID_HANDLE_VALUE)
    {
        CloseHandle(hFile);
        Msg("未发现视频列表, 正在从服务器下载...\r\n");
        if (!DownloadXML((string *)TotalXML))
        {
            return;
        }
        EnableWindow(hWndButton, true);
    }
else
{
    Msg("正在检测列表文件更新...\r\n");
    dwDataSize = GetFileSize(hFile, NULL);
    HttpMessenger *hm_GetTotalXML=new HttpMessenger(server_addr);
    if (!hm_GetTotalXML->CreateConnection())
    {
        Msg("检测更新时, 连接服务器失败, 错误日志:%s\r\n",
            (char *)hm_GetTotalXML->GetErrorLog().c_str());
        delete[] hm_GetTotalXML;
        return;
    }
    if (!hm_GetTotalXML->CreateAndSendRequest("GET",
        "/mov/xml/Total.xml", server_addr, NULL, true))
    {
        Msg("检测更新时, 发送请求失败, 错误日志:\r\n%s",
            hm_GetTotalXML->GetErrorLog().c_str());
        delete[] hm_GetTotalXML;
        return;
    }
    if (dwDataSize==hm_GetTotalXML->m_ContentLength)
    {
        delete []hm_GetTotalXML;
        Msg("列表文件为最新, 继续使用.\r\n");
        char *lpDataBuffer=new char[dwDataSize];
        if (!ReadFile(hFile, lpDataBuffer, dwDataSize, &dwRead, NULL))
        {
            Msg("Read XML File Error: %d\r\n", GetLastError());
            return;
        }
        *(string *)TotalXML+=lpDataBuffer;
    }
}

```

```

        delete[] lpDataBuffer;
        CloseHandle(hFile);
    }
    else
    {
        delete []hm_GetTotalXML;
        CloseHandle(hFile);
        Msg("电影列表有更新，从服务器下载.\r\n");
        if (!DownloadXML((string *)TotalXML))
        {
            return;
        }
    }
}
EnableWindow(hwndButton,true);
Msg("初始化完成.\r\n可输入视频名，演员，简介，分类等各种关键字搜索或直接复制介绍页面的网址进行提取\r\n");
return;
}

```

### 5.3.2 搜索并在列表框显示符合要求的视频

这部份是在用户输入关键字或链接，点击捕获按钮后发生，并且这应该是一个线程函数，因为整个搜索过程可能会耗费很多时间，如果直接在主线程中执行，会导致界面长时间停止响应。还应该提供让用户手动停止搜索过程的功能，这个功能应该让“捕获”按钮在被点击后修改为“停止捕获”来完成。其余的功能包括：

- (1) 处理用户输入：如果用户输入的是 URL 链接，则取出链接中的视频 ID 进入下一步，不是 URL 则直接进入下一步。
- (2) 按关键字获取相关视频 ID：根据关键字从 Total.xml 文件中获取所有包含此关键字的视频 ID。
- (3) 向服务器请求此视频相关信息：使用获取到的视频 ID 向服务器发送获取这个 ID 的视频的请求，接收服务器返回的数据。
- (4) 处理服务器返回的信息后显示：对服务器返回的数据进行处理，提取出想要的部份，并显示在列表框中。

完整的实现代码如下所示：

```

void SearchVideoKeyWord(void *txml)
{
    int nTextLen=0,start_pos=-1,end_pos=-1;
    char *szInput,*szLink;
    string strMovKeyWords;
    nTextLen=GetWindowTextLength(hwndEdit);
    if (nTextLen<4)

```

```

{
    MessageBoxA(hwnd, "为了搜索结果的准确性, 请多输入几个字.", 0, 0);
    return;
}
if (nTextLen>200)
{
    MessageBoxA(hwnd, "为了搜索到丰富资源, 请勿输入过多的关键字.", 0, 0);
    return;
}
szInput=new char[nTextLen+2];
ZeroMemory(szInput,nTextLen+2);
GetWindowText(hwndEdit,szInput,nTextLen+2);
strMovKeyWords+=szInput;
if (strMovKeyWords.find("http://")!=string::npos||
    strMovKeyWords.find("HTTP://")!=string::npos)
{
    start_pos=strMovKeyWords.find("html?info=")+10;
    end_pos=strMovKeyWords.find("&",start_pos);
    if (end_pos==string::npos)
    {
        end_pos=strMovKeyWords.length();
    }
    szLink=new char[end_pos-start_pos+2];
    ZeroMemory(szLink,end_pos-start_pos+2);
memcpy(szLink,strMovKeyWords.c_str()+start_pos,end_pos-start_pos);
ptrGetVideoID((string *)txml,szLink);
strMovKeyWords.erase();
delete szLink;
return;
}
ptrGetVideoID((string*)txml,szInput);
strMovKeyWords.erase();
delete szInput;
return;
}
void ptrGetVideoID(string *TotalXML,char *moviename)
{
    //将按钮更改为停止
    SetWindowText(hwndButton,TEXT("停止捕获"));
    EnableWindow(hListView,false);
    b_Search=true;//搜索已经开始
    Msg("开始捕获...\r\n");
    long current_loc=0;
    int end_loc=0,forward_loc=0,right_loc=0,left_loc=0,last_loc=0;
    do
    {

```



```

end_loc=0; forward_loc=0; right_loc=0; left_loc=0;
current_loc=TotalXML->find(moviename,current_loc);

if (current_loc==string::npos)
{
    Msg("搜索完成.\r\n");
    b_Search=false;
    SetWindowText(hwndButton,TEXT("捕获"));
    EnableWindow(hListView,true);
    break;
}
if (!b_Search)
{
    Msg("搜索被中止.\r\n");
    SetWindowText(hwndButton,TEXT("捕获"));
    EnableWindow(hwndButton,true);
    EnableWindow(hListView,true);
    break;
}
right_loc=TotalXML->rfind(">",current_loc);
//关键字不在视频名中，则向前搜索
if (TotalXML->at(right_loc-1)!='a')
{
    left_loc=TotalXML->rfind("<",right_loc);
    forward_loc=TotalXML->rfind("<b>",left_loc);
    forward_loc+=3;
    //关键字在同一视频的名字和简介中可能多次出现，进行过滤
    if (forward_loc==string::npos||last_loc==forward_loc)
    {
        current_loc+=strlen(moviename);
        continue;
    }
    last_loc=forward_loc;
    end_loc=TotalXML->find("</b>",forward_loc);
    char *cmdid=new char[end_loc-forward_loc+2];
    ZeroMemory(cmdid,end_loc-forward_loc+2);
    memcpy(cmdid,(TotalXML->c_str()+forward_loc),end_loc-forward_loc);
    GetVideoSummary(cmdid);
    current_loc+=strlen(moviename);
}
else
{
    current_loc=TotalXML->find("<b>",current_loc);
    current_loc+=3;//strlen("<b>");
    if (current_loc==string::npos)
    {

```

```

        Msg("未找到电影ID\r\n");
        continue;
    }
    last_loc=current_loc;
    end_loc=TotalXML->find("</b>",current_loc);
    char *cmdid=new char[end_loc-current_loc+2];
    ZeroMemory(cmdid,end_loc-current_loc+2);
    memcpy(cmdid,(TotalXML->c_str()+current_loc),end_loc-current_loc);
    GetVideoSummary(cmdid);
}
} while (true);
}
void GetVideoSummary(char *videoid)
{
    string ResourcePath;
    HttpMessenger *hm_GetVideoSummary=new HttpMessenger(server_addr);
    if (!hm_GetVideoSummary->CreateConnection())
    {
        Msg("连接视频服务器失败, 错误日志:%s\r\n",
            (char *)hm_GetVideoSummary->GetErrorLog().c_str());
        delete[] hm_GetVideoSummary;
        return;
    }
    ResourcePath="/mov/";
    ResourcePath+=videoid;
    ResourcePath+="/film.xml";
    if (!hm_GetVideoSummary->CreateAndSendRequest("GET",
        (char *)ResourcePath.c_str(),server_addr))
    {
        Msg("向服务器发送请求失败, 错误日志:\r\n%s",
            hm_GetVideoSummary->GetErrorLog().c_str());
        delete[] hm_GetVideoSummary;
        return;
    }
    int start_pos=0,end_pos=0;
    if (hm_GetVideoSummary->m_ResponseText.length()==0)
    {
        Msg("服务器未返回数据.\r\n");
        return;
    }
    start_pos=hm_GetVideoSummary->m_ResponseText.find("<name>")+6;
    if (start_pos==string::npos)
    {
        Msg("服务器返回错误响应.\r\n");
        return;
    }
}

```

```

        end_pos=hm_GetVideoSummary->m_ResponseText.find("</name>");
        char *videoname=new char[end_pos-start_pos+2];
        ZeroMemory(videoname,end_pos-start_pos+2);
        memcpy(videoname,&hm_GetVideoSummary->m_ResponseText[start_pos],e
nd_pos-start_pos);
        start_pos=0;end_pos=0;
start_pos=hm_GetVideoSummary->m_ResponseText.find("<director>")+10;
        end_pos=hm_GetVideoSummary->m_ResponseText.find("</director>");
        char *videodirector=new char[end_pos-start_pos+2];
        ZeroMemory(videodirector,end_pos-start_pos+2);
        memcpy(videodirector,&hm_GetVideoSummary->m_ResponseText[start_po
s],end_pos-start_pos);
        start_pos=0;end_pos=0;
        start_pos=hm_GetVideoSummary->m_ResponseText.find("<actor>")+7;
        end_pos=hm_GetVideoSummary->m_ResponseText.find("</actor>");
        char *videoactor=new char[end_pos-start_pos+2];
        ZeroMemory(videoactor,end_pos-start_pos+2);
        memcpy(videoactor,&hm_GetVideoSummary->m_ResponseText[start_pos],
end_pos-start_pos);
        start_pos=0;end_pos=0;
start_pos=hm_GetVideoSummary->m_ResponseText.find("<channelid>")+11;
        end_pos=hm_GetVideoSummary->m_ResponseText.find("</channelid>");
        char *videochannel=new char[end_pos-start_pos+2];
        ZeroMemory(videochannel,end_pos-start_pos+2);
        memcpy(videochannel,&hm_GetVideoSummary->m_ResponseText[start_pos
],end_pos-start_pos);
        // Msg("视频名称:%s\r\n",videoname);
        AddListViewItems(hListView,totalresult,videoname,videodirector,vi
deoactor,videochannel,videoid);
        totalresult++;
        delete []videoname;
        delete []videoactor;
        delete []videochannel;
        delete []videodirector;
    }

```

### 5.3.3 获取下载地址

此时用户可以从搜索结果中选择需要的影片，当用户选中某部影片后应该开始获取该片的下载地址，获取下载地址从影片的第一集开始，并且遍历完该影片的所有剧集的下载地址，遍历函数也应该放在一个单独的线程中执行，同样要提供手动停止遍历的功能，这里也是使用修改“捕获”按钮为“停止抓取”来实现。

这里使用 C++ 标准模板库 (STL) 提供的链表模板来存放影片信息结构体，影片信息结构体由视频名称和视频下载地址两个成员结成，声明全局迭代器以方便随时对链表进行操作。定义如下：

```
typedef struct _movinfo
{
    string mov_name;
    string mov_link;
}movinfo;
list<movinfo> MovList;
list<movinfo>::iterator i_ML;
```

获取影片下载地址的实现过程如下：

```
void GetVideoAddress(void *videoid)
{
    string ResourcePath; int Episode=0;
    char c_Episode[20];char szName[256];
    char szFileName[512]; char szFormat[20];
    ListView_GetItemText(hListView,selected,0,szName,256);
    SetWindowText(hWndButton,TEXT("停止抓取"));
    EnableWindow(hListView,false);
    b_Search=true;//搜索已经开始
    MovList.clear();//清空电影列表
    _itoa_s(0,c_Episode,10);
    while(true)
    {
        HttpMessenger *hm_GetVideoAddress=new HttpMessenger(server_addr);
        if (!hm_GetVideoAddress->CreateConnection())
        {
            Msg("遍历下载地址时，连接服务器失败，错误日志:%s\r\n",
                (char *)hm_GetVideoAddress->GetErrorLog().c_str());
            delete[] hm_GetVideoAddress;
            return;
        }
        ResourcePath="/xy_new.asp?a=";
        ResourcePath+=c_Episode;
        ResourcePath+="&b=";
        ResourcePath+=(char *)videoid;
        ResourcePath+="&time=2012-11-2822:35:41";
        if (!hm_GetVideoAddress->CreateAndSendRequest("GET",
            (char *)ResourcePath.c_str(),server_addr))
        {
            Msg("获取下载地址的请求发送失败，错误日志:\r\n%s",
                hm_GetVideoAddress->GetErrorLog().c_str());
            delete[] hm_GetVideoAddress;
            return;
        }
        if (hm_GetVideoAddress->m_ContentLength==0)
        {
            Msg("此视频还没有提供下载.\r\n");
        }
    }
}
```

```

        break;
    }
    int start_pos=0,end_pos=0;
    start_pos=hm_GetVideoAddress->m_ResponseText.find("|||")+3;
    end_pos=hm_GetVideoAddress->m_ResponseText.find("|||",start_pos);
    char *videoaddr=new char[end_pos-start_pos+2];
    ZeroMemory(videoaddr,end_pos-start_pos+2);
    memcpy(videoaddr,&hm_GetVideoAddress->m_ResponseText[start_pos],end_pos-start_pos);
    Episode=atoi(c_Episode)+1;
    ZeroMemory(szFileName,512);
    ZeroMemory(szFormat,20);
    start_pos=hm_GetVideoAddress->m_ResponseText.find_last_of(".");
    memcpy(szFormat,&hm_GetVideoAddress->m_ResponseText[start_pos],end_pos-start_pos);
    sprintf_s(szFileName,"%s_第%d集%s",szName,Episode,szFormat);
    Msg("%s\r\n%s\r\n",szFileName,videoaddr);
    movinfo mMovie;
    mMovie.mov_link=videoaddr;
    mMovie.mov_name=szFileName;
    MovList.push_back(mMovie);
    if ((end_pos+3)>=hm_GetVideoAddress->m_ResponseText.length())
    {
        if (MovList.size()==1)
        {
            i_ML=MovList.begin();
            i_ML->mov_name.erase();
            i_ML->mov_name=szName;
            i_ML->mov_name+=szFormat;
        }
        Msg("已经遍历完所有地址...你可以使用迅雷下载进行选择性下载.\r\n");
        Msg("再次在此影片名称上右击可直接播放哦...\r\n\r\n");
        break;}
    if (!b_Search)
    {
        Msg("遍历过程被中止.\r\n\r\n");
        break;
    }
    ZeroMemory(c_Episode,20);
    memcpy(c_Episode,&hm_GetVideoAddress->m_ResponseText[end_pos+3],hm_GetVideoAddress->m_ResponseText.length()-end_pos);
    delete[] hm_GetVideoAddress;
}
SetWindowText(hwndButton,TEXT("捕获"));
EnableWindow(hwndButton,true);
EnableWindow(hListView,true);

```

```
b_Search=false;//搜索已经停止
}
```

#### 5.3.4 调用迅雷进行下载

此时在链表中已经有了用户所选择的影片的所有下载地址和影片名，用户可以直接播放，也可以使用迅雷下载，用户也可以从提示信息中复制所显示的下载地址来自行决定如何处理。另外，从 5.1.2 小节的描述可知，本工具所实现的 HTTP 通信类是可以用于直接下载视频文件的，但考虑到下载的稳定性和易于管理等，工具自身并没有实现直接下载视频文件的功能，如果其它开发者有兴趣将直接下载功能加入工具中，可以参照本工具下载播放器组件时的代码，那是一个完整的使用 HttpMessenger 类下载大文件的示例，对于断点续传功能只需要修改 HTTP 请求头即可。

对于调用迅雷下载，这里是直接调用迅雷所提供的 COM 接口，对于未安装迅雷的用户应该给出提示信息。迅雷 COM 接口的调用需要导入迅雷提供的动态链接库文件，并使用正确的命名空间。

```
#import "G:\\\\Independent\\Thunder 7\\BHO\\ThunderAgent.dll"
using namespace ThunderAgentLib;
```

完整的迅雷调用代码如下：

```
bool CallThunder()
{
    HKEY hKEY;
    LPCTSTR tdRegPath = "Software\\Thunder
Network\\ThunderOem\\Thunder_backwnd\\";
    if(ERROR_SUCCESS!=RegOpenKeyEx( HKEY_LOCAL_MACHINE, tdRegPath, 0,
KEY_READ, &hKEY))
    {
        MessageBox(NULL,"你的机器上可能没有安装，或安装的是阉割版迅雷。", "未找到迅雷",0);
        return false;
    }
    LPBYTE tdPath = new BYTE[MAX_PATH*2];
    DWORD cbData = MAX_PATH*2;
    DWORD type = REG_SZ;
    if(ERROR_SUCCESS!=RegQueryValueEx(hKEY, "instdir", NULL, &type,
tdPath, &cbData))
    {
        MessageBox(NULL,"无法获取迅雷安装目录。", "未找到迅雷",0);
        return false;
    }
    delete tdPath;
    RegCloseKey(hKEY);
    HRESULT hr = CoInitialize(0);
```

```

IAgent *pAgent = NULL;
hr = CoCreateInstance(__uuidof(Agent), NULL, CLSCTX_INPROC_SERVER,
__uuidof(IAgent), (void**) &pAgent);
if (hr!=S_OK)
{
    Msg("调用迅雷的COM接口失败\r\n");
}
if(!MovList.empty())
{
    for (i_ML=MovList.begin();i_ML!=MovList.end();i_ML++)
    {
        pAgent->AddTask(i_ML->mov_link.c_str(),i_ML->mov_name.c_str(),_T(
""),_T(""),_T(""),1,0,1);
    }
    pAgent->CommitTasks();
    pAgent->Release();
    MovList.clear();
    CoUninitialize();
}
else
{
    MessageBox(hwnd,"请先选择你要下载的电影!", "未选择电影",0);
    return false;
}
return 0;
}

```

### 5.3.5 直接播放

考虑到并不是所有用户都会使用直接播放功能，以及保持工具的小巧便于传播，并没有将播放器组件与视频下载工具进行捆绑，而是用户确实需要在线直接播放视频，在首次播放时会自动从互联网上下载播放器，并自动完成解压和自动调用，整个过程不需要用户干预，只需要等待一到两分钟即可。

这里将生成 M3U 格式的播放列表文件，并自动调用播放器 MPlayer，进行播放，为了方便操作还附加了其图形界面 SMPlayer。

MPlayer 是一款开源的多媒体播放器，以 GNU 通用公共许可证发布。可在各主流操作系统使用，例如 Linux 和其他类 Unix 操作系统、微软 Windows 系统及苹果电脑的 Mac OS X 系统。MPlayer 是建基于命令行界面，在各操作系统可选择安装不同的图形界面。Smplayer 是一个基于 Qt 库，用 Mplayer 作为后端的媒体播放器。虽然是 Qt 程序，但是在 Gnome 下兼容良好。SMPlayer 有一个特色功能是它能记住你播放任何文件时候的设置。

经过测试，这样的组合能够提供比学校提供的视频插件更好的体验，并且对于上百集的视频，用户可以将生成的播放列表保存下来，以后可以使用 SMPlayer

来打开，直接播放，而不必每次都重新获取下载地址。

播放器的下载和调用代码如下：

```
bool CreateListAndPlay()
{
    HANDLE playerFile=CreateFile(".\\smplayer\\smplayer.exe",
        GENERIC_READ,0,0,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,0);
    if (playerFile!=INVALID_HANDLE_VALUE)
    {
        CloseHandle(playerFile);
    }
    else
    {
        if (MessageBox(hwnd,"未找到播放器组件.自动下载并安装?","没有播放器",1)==1)
        {
            _beginthread(DownloadAndInstallPlayer,0,NULL);
            return true;
        }
        else{return false;}
    }
    string m3u("#EXTM3U\n# Playlist created by CampusNetworkVideoHunter
1.0\n");
    string cmdline;
    if(!MovList.empty())
    {
        for (i_ML=MovList.begin();i_ML!=MovList.end();i_ML++)
        {
            m3u+="#EXTINF:0, ";
            m3u+=i_ML->mov_name.c_str();
            m3u+="\n";
            m3u+=i_ML->mov_link.c_str();
            m3u+="\n";
        }
        HANDLE hFile=NULL;
        DWORD dwWrite=0;
        hFile=CreateFile("VideoList.m3u",GENERIC_WRITE,0,0,CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL,0);
        if (hFile==INVALID_HANDLE_VALUE)
        {
            Msg("列表正在被使用，请关闭播放器后重试.\r\n");
        }
        WriteFile(hFile,(char
*)m3u.c_str(),m3u.length(),&dwWrite,NULL);
        MovList.clear();
        CloseHandle(hFile);
        unsigned long pathsize=GetCurrentDirectory(0,NULL);
```



```

        char *path=new char[pathsize];
        GetCurrentDirectory(pathsize,path);
        cmdline+=".\smplayer\smplayer.exe -minigui \";
        cmdline+=path;
        cmdline+="\\VideoList.m3u\"";
        Msg("提示:播放器界面按Ctrl+L打开播放列表.\r\n");
        STARTUPINFO hs_StartInfo;
        PROCESS_INFORMATION hs_ProcessInformation;
        memset(&hs_StartInfo,0,sizeof(STARTUPINFO));
        memset(&hs_ProcessInformation,0,sizeof(PROCESS_INFORMATION));
        CreateProcess(NULL,(char
*)cmdline.c_str(),NULL,NULL,1,0,NULL,NULL,&hs_StartInfo,&hs_ProcessIn
formation);
        delete [] path;
    }
    else
    {
        MessageBox(hwnd,"请先选择你要播放的电影!", "未选择电影",0);
        return false;
    }
    return true;
}

void DownloadAndInstallPlayer(void *temp)
{
    if (b_Downloading)
    {
        Msg("正在下载,请稍候...\r\n");
        return;
    }
    b_Downloading=true;
    Msg("开始下载播放器组件...\r\n");
    HttpMessenger *hm_DownloadPlayer=new
HttpMessenger("cwnu-campus-network-video-hunter.googlecode.com");
    if (!hm_DownloadPlayer->CreateConnection())
    {
        Msg("下载播放器组件时,连接服务器失败,错误日志:%s\r\n",
            (char *)hm_DownloadPlayer->GetErrorLog().c_str());
        delete[] hm_DownloadPlayer;
        return;
    }

    if (!hm_DownloadPlayer->CreateAndSendRequest("GET",
"/files/SMPlayer.exe","cwnu-campus-network-video-hunter.googlecode.co
m",NULL,false,"SMPlayer.exe"))
    {
        Msg("下载播放器组件时,错误日志:\r\n%s",

```

```

        hm_DownloadPlayer->GetErrorLog().c_str());
    delete[] hm_DownloadPlayer;
    return;
}
Msg("下载完成...开始安装...\r\n");
b_Downloading=false;
STARTUPINFO hs_StartInfo;
PROCESS_INFORMATION hs_ProcessInformation;
memset(&hs_StartInfo,0,sizeof(STARTUPINFO));
memset(&hs_ProcessInformation,0,sizeof(PROCESS_INFORMATION));
if (!CreateProcess(NULL,"SMPlayer.exe",NULL,NULL,1,0,NULL,NULL,
    &hs_StartInfo,&hs_ProcessInformation))
{
    Msg("安装时发生错误, 错误代码:%ld\r\n",GetLastError());
    DeleteFile("SMPlayer.exe");
    Msg("这可能由杀毒软件导致, 可关闭杀毒软件后重试。安装取消...\r\n");
    return;
}
WaitForSingleObject(hs_ProcessInformation.hProcess,INFINITE);
DeleteFile("SMPlayer.exe");
Msg("安装完成...现在可以直接播放影片了...('?ω')\r\n");
}

```

## 5.4 最终实现效果

至此，所有的下载相关的功能都已完成，至于域名补丁和访问网页的代码请参见完整源代码文件。

### 5.4.1 Aero 效果和“关于程序”界面的 GDI+图形绘制

在 Windows 7 下启用 Aero 效果和“关于程序”界面的 GDI+图形绘制效果。

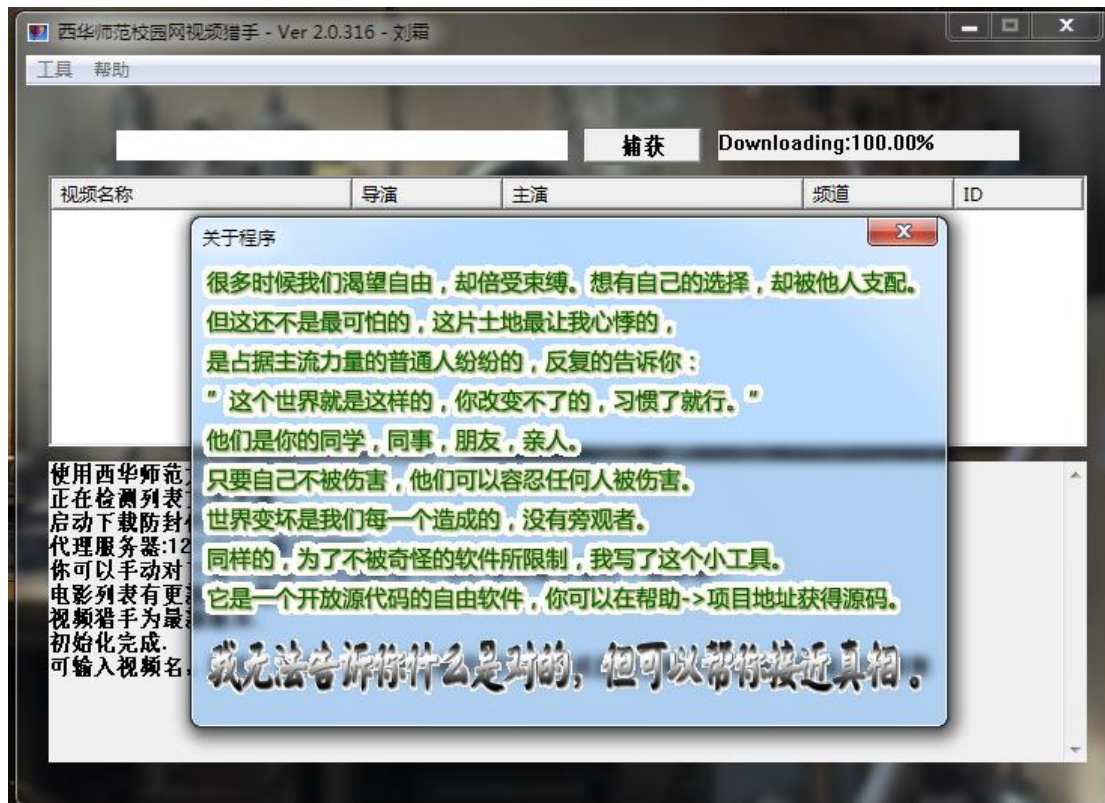


图 5.1 Aero 效果和“关于程序”界面的 GDI+图形绘制

#### 5.4.2 在 Windows XP 下初始下载 Total.xml 及“设置服务器地址”功能的演示

“设置服务器地址”功能允许其它学校的同学同样可以在简单的设置过后使用此工具进行下载。



图 5.2 自定义服务器地址和学校名称

### 5.4.3 从搜索到获取下载地址并下载播放器进行观看的演示

这里以公开课为关键字进行搜索，并选择感兴趣的影片进行地址获取，再下载并调用 Smplayer 进行播放。

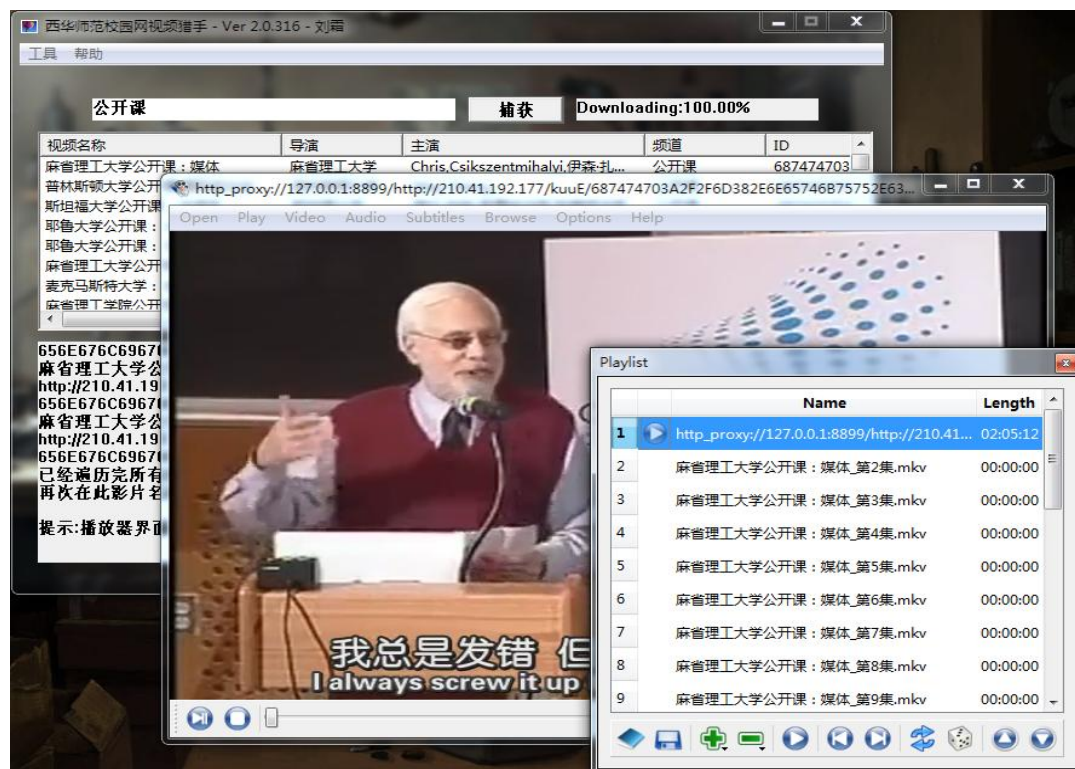


图 5.3 搜索影片并进行播放

### 5.4.4 批量下载及域名补丁效果

演示调用迅雷进行影片筛选下载及使用域名补丁功能后，直接使用 IE 访问。



图 5.4 迅雷下载及直接访问

## 第六章 测试

### 6.1 西华师范大学本校内的测试

在修复完我自己发现的所有 BUG，并完成了细节优化后，向西华师范大学的同学放出了此工具的下载地址，截止 2013 年 5 月 10 日，仅 Google Code 一处的单方下载统计数据就有 5000 多次，经过同学们的测试，校园网视频下载工具的确能绝大多数同学的计算机上正常稳定的运行，下载和播放功能都能正常使用，并且不用进行其它额外的设置操作。

### 6.2 其它院校的同学的测试

工具很快传播到了其它院校，包括湖南师范大学，西安石油大学，四川科技职业学院，河南理工大学，安徽工程大学等数十所高校，经过这些院校同学的使用和热心反馈，证实本校园网视频下载工具的确能适用于其它基于“光音网视高校云视频”系统的搭建的校园视频网站，在此再次感谢这些学校同学的测试和反馈。

## 第七章 总结

### 7.1 关于开源

#### 7.1.1 开源的原因

为了让此工具一直有人维护，并且可以正常使用。再加之目前的所有杀毒软件误杀率极高，原因之一是所有病毒和木马的作者在放出程序前都会严格测试，保证其程序不被常见的杀毒软件所查杀，而大多数小工具并不会做这种测试，开放源代码是证明自身不包含恶意代码的最直接方式。

#### 7.1.2 开源的方式

项目代码托管于 Google Code。

项目地址：

<http://code.google.com/p/cwnu-campus-network-video-hunter/>

项目源代码在线查看：

<http://code.google.com/p/cwnu-campus-network-video-hunter/source/browse/trunk/CampusNetworkVideoHunter/>

完整源代码检出：



```
svn
```

```
checkout http://cwnu-campus-network-video-hunter.googlecode.com/svn/trunk/  
cwnu-campus-network-video-hunter-read-only
```

## 7.2 总结

整个校园网视频下载工具从可行性分析到设计，再到最后实现，一切都还算轻松，对我来说并没有太困难的东西。从最初的命令行版的小工具，到可以被大家所使用的实用工具，离不开同学们的支持，不然我也没有耐心把这个工具写完，可能到现在也只是一个方便我自己使用的 Python 脚本。工具还有很多需要改进的地方，也欢迎用户更多的提出意见，以帮助工具更好的维护下去。

## 参考文献：

- [1]Mario Hewardt , Daniel Pravat. Advanced Windows Debugging[M].  
US:Pearson Education, inc., 2008
- [2]吴翰清. 白帽子讲 Web 安全[M]. 北京：电子工业出版社. 2012

## 致谢