原

**关于获取多个屏幕分辨率以及进行一些设置**

2017年02月04日 23:13:04 [swartz\_lubel](https://me.csdn.net/swartz_lubel) 阅读数：3157 标签： [qt](http://so.csdn.net/so/search/s.do?q=qt&t=blog) 更多

个人分类： [Qt](https://blog.csdn.net/swartz_lubel/article/category/6693940)

[Windows](https://www.baidu.com/s?wd=Windows&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)下提示显示器信息主要通过两个函数实现。一个是EnumDisplayDevices()， 另一个是EnumDisplayMonitors().  
EnumDisplayDevices()枚举所有显示设备，而EnumDisplayMonitors枚举的是所有显示器。  
显示设备和显示器不一样，比如显卡算显示设备，但是不是显示器。  
EnumDisplayMonitors()还会枚举出不可见的伪显示器，如果只是想得到实际的显示器数目的话可以用GetSystemMetrics(SM\_CMONITORS)，该函数不包括虚拟显示器。

MonitorInfoex 和MonitorInfo 在这两个结构中保存着相应显示器的相关信息,如坐标、是否为主显示器等

GetMonitorInfo ( ) 　取得指定显示器的相关信息,如物理显示区大小等  
MonitorFromPoint ( ) 　取得指定点所在的显示器句柄  
MonitorFromRect ( ) 　取得指定矩形所在的显示器句柄  
MonitorFromWindow( ) 　取得指定窗口所在的显示器句柄  
MonitorEnumProc( ) 当应用程序调用EnumDisplayMonitors ( )查询显示器个数时,系统自动为每一个显示器调用一次该函数。应用程序可以依此判断显示器的个数、位置及显示区域的大小等信息

函数名：MonitorEnumProc（）  
输入参数：HMONITOR hMonitor ---显示器句柄  
          HDC hdcMonitor  ----显示器DC句柄  
        LPRECT lprcMonitor-----  
        LPARAM dwData-----EnumDisplayMonitors传来的数据  
返回：bool  
功能：若返回为真，EnumDisplayMonitors继续枚举，  
      若返回为假，EnumDisplayMonitors停止枚举，从而获得显示器信息

MonitorAdapter.h

1. #include <atlstr.h>
2. #include <vector>
3. #include <WinDef.h>
4. #include <tchar.h>
5. using std::vector;
7. using namespace std;
9. #define MAX\_MONITOR\_NAME 256
11. static std::vector<HMONITOR> g\_hMonitorGroup;
13. // 显示器模式信息
14. typedef struct MonitorModeInfo\_t
15. {
16. unsigned int m\_nWidth;
17. unsigned int m\_nHeight;
19. MonitorModeInfo\_t(int nWidth, int nHeight) : m\_nWidth(nWidth), m\_nHeight(nHeight) {}
20. }MonitorModeInfo;
22. // 显示器信息
23. struct MonitorInfo
24. {
25. TCHAR szDevice[MAX\_MONITOR\_NAME]; // 显示器名称
26. std::vector<MonitorModeInfo> m\_vecModeInfo; // 当前名称的显示器支持的分辨率模式
27. };
29. typedef std::vector<MonitorInfo> VEC\_MONITORMODE\_INFO; // 所有的显示器信息

32. class MonitorAdapter
33. {
34. public:
35. MonitorAdapter();
36. ~MonitorAdapter();
37. // 回调函数
38. static int CALLBACK MonitorEnumProc(HMONITOR hMonitor,
39. HDC hdc,
40. LPRECT lpRMonitor,
41. LPARAM dwData);
43. // 得到所有显示器的名称
44. void GetAllMonitorName(VEC\_MONITORMODE\_INFO& m\_vecMonitorListInfo);
46. // 得到所有显示器的模式
47. void GetAllDisplayMode(VEC\_MONITORMODE\_INFO& m\_vecMonitorListInfo);
49. //得到屏幕当前分辨率
50. void GetCurrentReselotion(int& nWidth,int& nHeight,int& nFreq,int& nBits);
52. //根据屏幕ID取获取屏幕的对应分辨率
53. void GetCurrentReselotion(LPCWSTR lpszDeviceName, int& nWidth, int& nHeight, int& nFreq, int& nBits);
55. //修改分辨率
56. int ChangMonitorReselotion(HMONITOR hMonitor,const int nWidth,const int nHight,const int nFre,const int nColorBits);
57. };

MonitorAdapter.cpp

1. #include "stdafx.h"
2. #include "MonitorAdapter.h"
4. MonitorAdapter::MonitorAdapter(void)
5. {
6. }
8. MonitorAdapter::~MonitorAdapter(void)
9. {
10. }

13. int CALLBACK MonitorAdapter::MonitorEnumProc(HMONITOR hMonitor,
14. HDC hdc,
15. LPRECT lpRMonitor,
16. LPARAM dwData)
17. {
18. g\_hMonitorGroup.push\_back(hMonitor);
19. return 1;
20. }

23. // 得到所有显示器的名称
24. void MonitorAdapter::GetAllMonitorName(VEC\_MONITORMODE\_INFO& vecMonitorListInfo)
25. {
26. g\_hMonitorGroup.clear();
27. ::EnumDisplayMonitors(NULL, NULL, MonitorEnumProc, 0);
29. //vector<HMONITOR>::iterator ithMoniter = g\_hMonitorGroup.begin();
30. for(int i = 0; i < g\_hMonitorGroup.size();i++)
31. {
32. MONITORINFOEX mixTemp;
33. memset(&mixTemp, 0, sizeof(MONITORINFOEX));
34. mixTemp.cbSize = sizeof(MONITORINFOEX);
36. GetMonitorInfo(g\_hMonitorGroup[i], &mixTemp);
37. VEC\_MONITORMODE\_INFO::iterator itBeg = vecMonitorListInfo.begin();
38. VEC\_MONITORMODE\_INFO::iterator itEnd = vecMonitorListInfo.end();
39. for(; itBeg != itEnd; ++itBeg)
40. {
41. if( 0 == \_tcscmp(mixTemp.szDevice, itBeg->szDevice))
42. {
43. break;
44. }
45. }
47. //没有在列表中找到,则需要添加
48. if (itBeg == itEnd)
49. {
50. MonitorInfo tmpMonitorInfo;
51. \_tcscpy\_s(tmpMonitorInfo.szDevice, sizeof(tmpMonitorInfo.szDevice), mixTemp.szDevice);
52. vecMonitorListInfo.push\_back(tmpMonitorInfo);
53. }
54. }
55. }

58. // 得到所有显示器的模式
59. void MonitorAdapter::GetAllDisplayMode(VEC\_MONITORMODE\_INFO& vecMonitorListInfo)
60. {
61. GetAllMonitorName(vecMonitorListInfo);
63. bool bRetVal;
64. DEVMODE devmode;
66. VEC\_MONITORMODE\_INFO::iterator itBeg = vecMonitorListInfo.begin();
67. VEC\_MONITORMODE\_INFO::iterator itEnd = vecMonitorListInfo.end();
68. for (NULL; itBeg != itEnd; ++itBeg)
69. {
70. int iMode = 0;
71. do
72. {
73. bRetVal = ::EnumDisplaySettings(itBeg->szDevice, iMode, &devmode);
74. iMode++;
75. if (bRetVal)
76. {
77. bool bFind = false;
79. vector<MonitorModeInfo>::iterator itBeg\_Mode = itBeg->m\_vecModeInfo.begin();
80. vector<MonitorModeInfo>::iterator itEnd\_Mode = itBeg->m\_vecModeInfo.end();
81. for (NULL; itBeg\_Mode != itEnd\_Mode; ++itBeg\_Mode)
82. {
83. // 如果已经在列表中找到,则结束本次循环
84. if ((itBeg\_Mode->m\_nWidth == devmode.dmPelsWidth) && (itBeg\_Mode->m\_nHeight == devmode.dmPelsHeight))
85. {
86. bFind = true;
87. break;
88. }
90. // 插入数据时,从 大到小排列 (按windows 分辨率设置,优先比较 宽)
91. if (
92. (itBeg\_Mode->m\_nWidth < devmode.dmPelsWidth) ||
93. ((itBeg\_Mode->m\_nWidth == devmode.dmPelsWidth) && (itBeg\_Mode->m\_nHeight < devmode.dmPelsHeight))
94. )
95. {
96. break;
97. }
98. }
100. if(!bFind)
101. {
102. if (itBeg\_Mode == itEnd\_Mode)
103. {
104. itBeg->m\_vecModeInfo.push\_back(MonitorModeInfo(devmode.dmPelsWidth, devmode.dmPelsHeight));
105. }
106. else
107. {
108. itBeg->m\_vecModeInfo.insert(itBeg\_Mode, MonitorModeInfo(devmode.dmPelsWidth, devmode.dmPelsHeight));
109. }
110. }
111. }
112. }
113. while (bRetVal);
114. }
115. }

118. int MonitorAdapter::ChangMonitorReselotion(HMONITOR hMonitor,const int nWidth,const int nHight,const int nFre,const int nColorBits)
119. {
120. if ( NULL == hMonitor )
121. {
122. return -1;
123. }
124. MONITORINFOEX mi;
125. mi.cbSize = sizeof(mi);
126. GetMonitorInfo( hMonitor , &mi);
127. DEVMODE DeviceMode;
128. ZeroMemory(&DeviceMode, sizeof(DEVMODE));
129. DeviceMode.dmSize = sizeof(DEVMODE);
131. BOOL bFlag = TRUE;
132. bFlag = EnumDisplaySettings(mi.szDevice, ENUM\_CURRENT\_SETTINGS, &DeviceMode);
133. if ( bFlag != TRUE )
134. {
135. return -1;
136. }
137. if (DeviceMode.dmPelsWidth == nWidth && DeviceMode.dmPelsHeight == nHight )
138. {
139. return 0;
140. }
141. DeviceMode.dmDisplayFlags = 0;
142. DeviceMode.dmPelsWidth= nWidth;
143. DeviceMode.dmPelsHeight = nHight;
145. DeviceMode.dmFields = DM\_PELSWIDTH | DM\_PELSHEIGHT | DM\_BITSPERPEL | DM\_DISPLAYFREQUENCY ;
147. int nRet = ChangeDisplaySettingsEx(mi.szDevice, &DeviceMode, NULL, CDS\_GLOBAL | CDS\_NORESET | CDS\_UPDATEREGISTRY, NULL);
148. if (DISP\_CHANGE\_BADMODE == nRet)
149. {
150. ChangeDisplaySettingsEx(mi.szDevice, &DeviceMode, NULL, CDS\_GLOBAL | CDS\_NORESET | CDS\_UPDATEREGISTRY, NULL);
152. }
153. if ( DISP\_CHANGE\_SUCCESSFUL == nRet )
154. {
155. return 0;
156. }
157. return -1;
158. }
160. void MonitorAdapter::GetCurrentReselotion(int& nWidth,int& nHeight,int& nFreq,int& nBits)
161. {
162. DEVMODE DeviceMode;
163. EnumDisplaySettings(NULL, ENUM\_CURRENT\_SETTINGS, &DeviceMode);
164. nWidth = DeviceMode.dmPelsWidth;
165. nHeight = DeviceMode.dmPelsHeight;
166. nFreq = DeviceMode.dmDisplayFrequency;
167. nBits = DeviceMode.dmBitsPerPel;
168. }
170. void MonitorAdapter::GetCurrentReselotion(LPCWSTR lpszDeviceName, int& nWidth, int& nHeight, int& nFreq, int& nBits)
171. {
172. DEVMODE DeviceMode;
173. EnumDisplaySettings(lpszDeviceName, ENUM\_CURRENT\_SETTINGS, &DeviceMode);
174. nWidth = DeviceMode.dmPelsWidth;
175. nHeight = DeviceMode.dmPelsHeight;
176. nFreq = DeviceMode.dmDisplayFrequency;
177. nBits = DeviceMode.dmBitsPerPel;
178. }

test

2. #include "stdafx.h"
3. #include "MonitorAdapter.h"
5. int \_tmain(int argc, \_TCHAR\* argv[])
6. {
7. MonitorAdapter m\_monitorAdapter; //显示器
8. VEC\_MONITORMODE\_INFO vecMointorListInfo;
9. m\_monitorAdapter.GetAllDisplayMode(vecMointorListInfo);
11. int nWidth = 0, nHeight = 0, nFreq = 0, nBits = 0;
12. VEC\_MONITORMODE\_INFO::iterator itBeg = vecMointorListInfo.begin();
13. for (int i = 0; i < vecMointorListInfo.size();i++)
14. {
15. //得到当前显示器分辨率 刷新率 色位
16. m\_monitorAdapter.GetCurrentReselotion(itBeg->szDevice, nWidth, nHeight, nFreq, nBits);
17. itBeg++;
18. nWidth = 0, nHeight = 0, nFreq = 0, nBits = 0;
19. }
21. return 0;
22. }