

第7讲 | 如何建立一个Windows窗体？

2018-06-09 蔡能

从0开始学游戏开发

[进入课程 >](#)



今天，我要跟你分享开发 Windows 游戏的第一步，建立窗体。

上一节，我讲解 Python 和 C++ 的编译器，以及它们各自对应的 IDE 该如何选择，并且测试了 C/C++ 的运行，编译了一个 Lua 静态库。准备工作基本上算是完成了。

如果你有一些编程功底，应该知道建立 Windows 的窗体所需的一些基础知识。如果你经验稍丰富一些，还应该知道 Delphi、C++Builder、C# 等等。这些工具都可以帮助你非常方便地做出一个空白窗体，但是这些窗体并没有游戏的绘图系统，所以它们只是“建立了一个标准窗体”而已。因此，虽然建立窗体是我们这一节的内容，但**我们要探讨的是，在窗体背后，Windows 系统做了什么。**

Windows 窗体由哪些部分构成？

我们常规意义上的 Windows 窗体，由下列几个部分组成。

菜单栏：位于标题栏的下面，包含很多菜单，涉及的程序所负责的功能不一样，菜单的内容也不一样。比如有些有文件菜单，有些就没有，有一些窗体甚至根本就没有菜单栏。

工具栏：位于菜单栏的下方，工具栏会以图形按钮的形式给出用户最常使用的一些命令。比如，新建、复制、粘贴、另存为等。

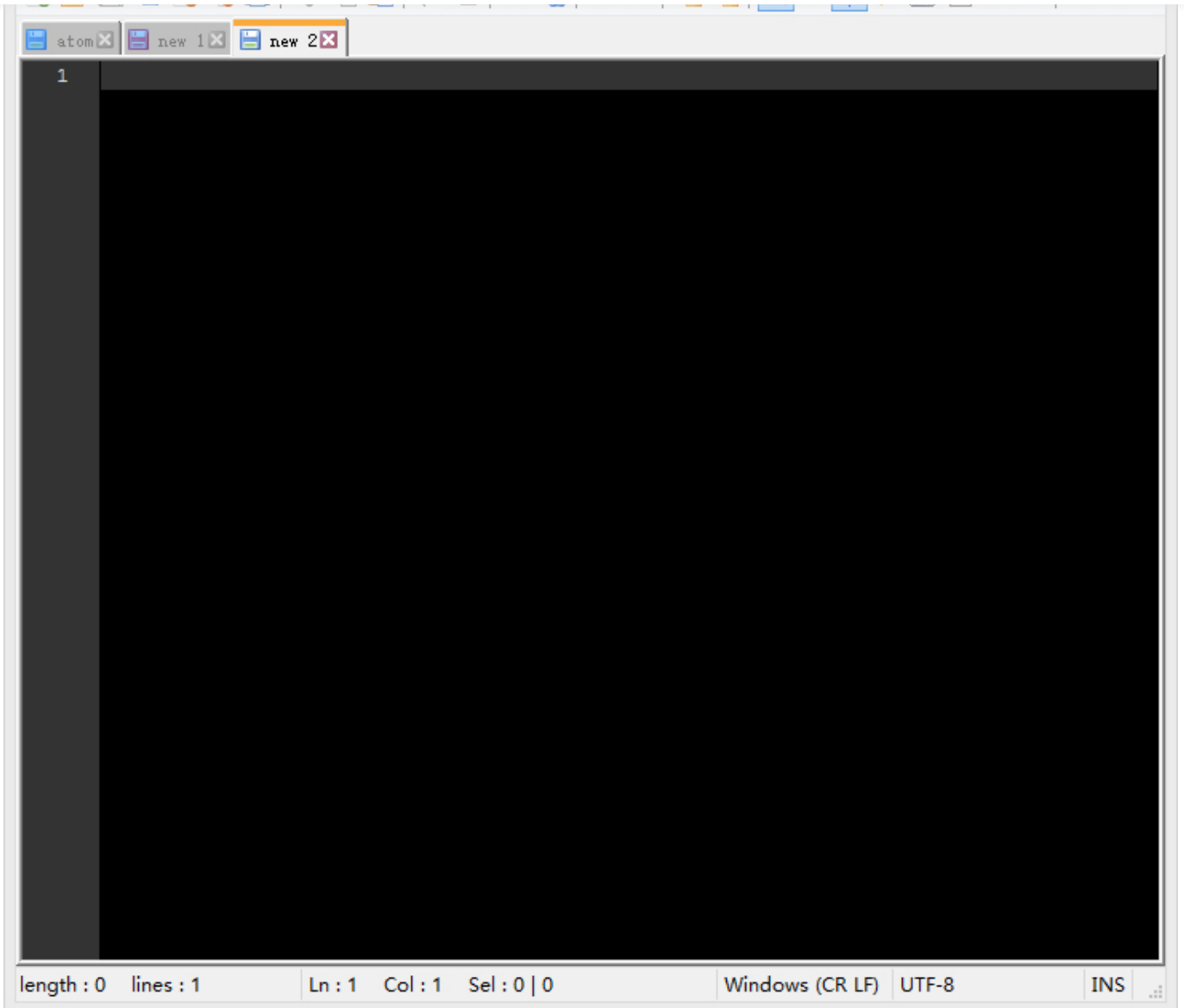
工作区域：窗体的中间区域。一般窗体的输入输出都在这里进行，如果你接触过 Windows 窗体编程，就知道在这个工作区域能做很多的事情，比如子窗体显示、层叠，在工作区域的子窗体内进行文字编辑等等。你可以理解成，游戏的图形图像就在此处显示。

状态栏：位于窗体的底部，显示运行程序的当前状态。通过它，用户可以了解到程序运行的情况。比如的，如果我们开发出的窗体程序是个编辑器的话，我按了一下 Insert 键，那么状态栏就会显示 Ins 缩写；或者点击到哪个编辑区域，会在状态栏出现第几行第几列这样的标注。

滚动条：如果窗体中显示的内容过多，不管横向还是纵向，当前可见的部分不够显示时，窗体就会出现滚动条，分为水平滚动条与垂直滚动条两种。

窗体缩放按钮：窗体的缩放按钮在右上角，在窗体编程中属于 System 类目。这些缩放按钮依次为最小化、最大化和关闭按钮。

我们来看一张标准的 Windows 窗体截图，这个软件名是 Notepad++。




这是 MSDN 上对于窗体结构的说明：

 复制代码

```
1 typedef struct tagWNDCLASSEX {
2     UINT        cbSize; // 结构体大小，等于 sizeof(WNDCLASSEX)
3     UINT        style;  // 窗体的风格
4     WNDPROC     lpfnWndProc; // 窗体函数指针
5     int         cbClsExtra; // 附加在窗体类后的字节数，初始化是零
6     int         cbWndExtra; // 附加在窗体实例化的附加字节数。系统初始化是零，如果一个应用程序使用
7     HINSTANCE   hInstance; // 该对象的实例句柄
8     HICON       hIcon;      // 该对象的图标句柄
9     HCURSOR     hCursor;    // 该对象的光标句柄
10    HBRUSH       hbrBackground; // 该对象的背景刷子
11    LPCTSTR      lpszMenuName; // 菜单指针
12    LPCTSTR      lpszClassName; // 类名指针
13    HICON        hIconSm;      // 与窗体关联的小图标，如果这个值为 NULL，那么就把 hIcon 转换为大
```


使用 C/C++ 编写 Windows 窗体

接下来，我将使用 C/C++ IDE 来编写代码，完成一个默认窗体的开发，并让它运行起来。

 复制代码

```
1 #include <windows.h>
2 LRESULT CALLBACK WindowProcedure(HWND, UINT, WPARAM, LPARAM);
3 char szClassName[ ] = "WindowsApp";
4 int WINAPI WinMain(HINSTANCE hThisInstance, HINSTANCE hPrevInstance, LPSTR lpszArgument
5
6 {
7     HWND hwnd;           /* 指向我们窗体的句柄 */
8     MSG messages;        /* 保存发往应用的消息 */
9     WNDCLASSEX wincl;    /* 前面详细介绍过的 WNDCLASSEX 结构的对象 */
10    wincl.hInstance = hThisInstance;
11    wincl.lpszClassName = szClassName;
12    wincl.lpfnWndProc = WindowProcedure;
13    wincl.style = CS_DBLCLKS;
14    wincl.cbSize = sizeof(WNDCLASSEX);
```

上述代码开始给 WNDCLASSEX 结构对象赋值。

 复制代码

```
1 /* 使用默认图标以及鼠标指针 */
2 wincl.hIcon = LoadIcon(NULL, IDI_APPLICATION);
3 wincl.hIconSm = LoadIcon(NULL, IDI_APPLICATION);
4 wincl.hCursor = LoadCursor(NULL, IDC_ARROW);
5 wincl.lpszMenuName = NULL; /* 没有菜单栏 */
6 wincl.cbClsExtra = 0;           /* 没有多余的字节跟在窗体类的后面 */
7 wincl.cbWndExtra = 0;
8 wincl.hbrBackground = (HBRUSH) GetStockObject(LTGRAY_BRUSH);
9 if(!RegisterClassEx(&wincl)) return 0;
```


代码在窗口过程调用函数的时候，将地址赋值给 lpfnWndProc，然后呼叫 RegisterClassEx(&wincl) 注册窗口类，系统就拥有了窗口过程函数的地址。如果注册失败，则返回 0。

```

2         szClassName,          /* 类名 */
3         "Windows App",        /* 窗体抬头标题 */
4         WS_OVERLAPPEDWINDOW, /* 默认窗体 */
5         CW_USEDEFAULT,        /* 让操作系统决定窗体对应 Windows 的 X 位置在哪里 */
6         CW_USEDEFAULT,        /* 让操作系统决定窗体对应 Windows 的 Y 位置在哪里 */
7         544,                  /* 程序宽度 */
8         375,                  /* 程序高度 */
9         HWND_DESKTOP,         /* 父窗体的句柄，父窗体定义为 Windows 桌面，HWND_DESKTOP 是
10        NULL,                  /* 没有菜单 */
11        hThisInstance,        /* 程序实例化句柄 */
12        NULL                  /* 指向窗体的创建数据为空 */
13    );
14    ShowWindow(hwnd, nFunsterStil);
15    /* 要显示窗体，使用的是 ShowWindow 函数 */
16    while(GetMessage(&messages, NULL, 0, 0))
17    {
18        TranslateMessage(&messages);
19        DispatchMessage(&messages);
20    }
21    return messages.wParam;
22 }

```

建立并显示窗体，在循环内将虚拟键消息转换为字符串消息，随后调度一个消息给窗体程序。

 复制代码


```

1 LRESULT CALLBACK WindowProcedure(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
2 {
3     switch (message)          /* 指向消息的句柄 */
4     {
5         case WM_DESTROY:
6             PostQuitMessage(0);
7             break;
8         default:
9             return DefWindowProc(hwnd, message, wParam, lParam);
10    }
11    }
12    return 0;
13 }

```

数指针并且通过函数指针调用函数对消息进行处理。

还有一个经常用到的函数就是 MoveWindow，就是移动已经建立的窗体。MoveWindow 函数用来改变窗口的位置和尺寸，如果窗体本身就按照计算机的屏幕对齐左上角，对于窗体内的子窗体，就对齐父窗体的左上角。

 复制代码

```
1 BOOL MoveWindow( HWND hWnd,/* 窗体句柄 */
2     int x, /* 窗体左上角起点 x 轴 */
3     int y, /* 窗体左上角起点 y 轴 */
4     int nWidth, /* 窗体宽度 */
5     int nHeight, /* 窗体高度 */
6     BOOL bRepaint = TRUE /* 是否重新绘制，如果是 true 系统会发送 WM_PAINT 到窗体，然后
7     );
```

MoveWindow 会给窗体发送 WM_WINDOWPOSCHANGING，WM_WINDOWPOSCHANGED，WM_MOVE，WM_SIZE 和 WM_NCCALCSIZE 消息。

类似的功能还有 SetWindowPos，SetWindowPos 功能更强大，可以设置更多的参数。


这是基本的使用 C/C++ 绘制 Windows 窗体的流程，也是标准的 Windows 窗体的创建和显示。在后续分享中，我也会使用 GDI 或者 GDI+ 来绘制一些的内容。

使用 Python 编写 Windows 窗体

说完了 C/C++ 系统编程编写的 Windows 窗体，接下来来看一下，如何使用 Python 来编写 Windows 窗体。

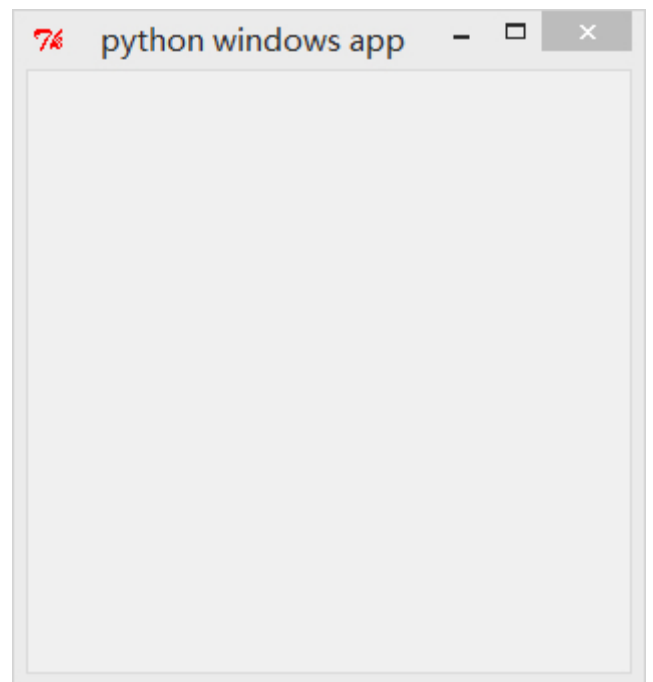
Python 的 Windows 窗体编程一般会使用默认的 Tinker 库。不过用别的窗体库也可建立一个窗体，比如 Python 版本的 QT 库或者 wxPython。

现在来看一下，使用默认的 Tinker 来建立一个窗体。

 复制代码

```
4  ws = root.winfo_screenwidth()
5  hs = root.winfo_screenheight()
6  x = (ws/2) - (w/2)
7  y = (hs/2) - (h/2)
8  root.geometry("%dx%d+%d+%d" % (w, h, x, y))
9
10 root = Tkinter.Tk(className='python windows app')
11 my_window(100, 100)
12 root.mainloop()
```

运行的结果是这样的。




我们可以看到左上角有一个 Tk 的标识，这是 Tinker 的默认图标。目前，我们只是建立了一个 Windows 的窗体，并不能直接编写游戏。除此之外，我们还必须要知道这些建立窗体的具体的细节。

不过，就像前面的文章所说，OpenGL 并不附带任何关联窗体的编程，所以如果你使用的是 OpenGL 的接口来编写代码，稍微修改一下，这些窗体就能成为游戏屏幕窗体。

游戏所有的内容都是在一个循环内完成的，即我们所有的绘图、线程、操作、刷新，都在一个大循环内完成，类似我们在前面看到的代码。

```
3     TranslateMessage(&messages);  
4     DispatchMessage(&messages);  
5 }
```

以及使用 Python 编写的代码的窗体中，也会看到一个循环函数：

 复制代码

```
1 root.mainloop()
```

在这个 while 循环中，消息的派发都在此完成。游戏也一样，我们所有游戏内的代码几乎都在循环内完成。你可以想象一个循环完成一个大的绘制过程，第二个循环刷新前一次绘制过程，最终类似电影一样，完成整个动画的绘制以及不间断的操作。

在建立 Windows 窗体的时候，程序会从入口函数 WinMain 开始运行，定义和初始化窗体类，然后将窗体类实例化，随后进行消息循环获取消息，然后将消息发送给消息处理函数，最后做出相应的操作。

小结

总结一下今天所说的内容，我们编写了一个标准的 Windows 窗体，在编写的过程中：

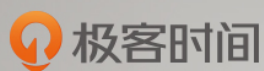
窗体的结构是在建立窗体之前就定义下来的；

所有长时间运行的程序，包括游戏，包括 Windows 本身都是一个大循环。我们在这个循环里做我们想做的事情，直到循环结束；

如果使用脚本语言的方式编写窗体，就不需要关心那么多的东西，只需要定义坐标、位置和窗体名称即可。

最后，给你留一道小思考题吧。

你经常会看到有一些游戏是需要全屏才能进行的。既然我们在这里建立了一个窗体，那请问你，全屏是怎么做到的呢？



从0开始学游戏开发

你的游戏开发入门第一课

蔡能

原网易游戏引擎架构师
资深游戏底层技术专家



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 第6讲 | 从0开始整理开发流程

下一篇 第8讲 | 如何区分图形和图像？

精选留言 (11)

写留言



Geek_King@...

2018-06-15

3

c++窗口那段程序，编译时需加上-mwindows或者-lgdi32,又或者在建工程的时候选择建立gui程序😄😄😄



呵呵

2018-06-09

3

要是能提供远嘛就好了，跟着后面敲太麻烦

展开 ∨



2018-06-11

老师，python能开发手机游戏吗，游戏引擎用的是pygame吗？

展开

作者回复: 目前没有成熟的python游戏引擎支持手机游戏开发。



GS

2018-07-28

python3 要用小写的tkinter

展开



宋恒公

2018-06-27

最大化，加无边框等于全屏

展开



作者回复: 这是一种方式，没错的



王大师

2018-06-13

老师，那手游的开发目前也都是C++咯？像简单游戏比如手机棋牌类游戏一般用什么语言？

作者回复: 不都是。原生开发，苹果会用obj c， 安卓java， 或者也有封装后生成的，比如cocos系列的语言

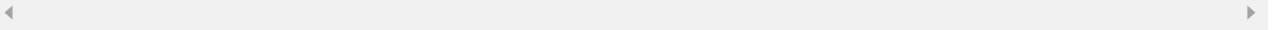


@浩

2018-06-09



作者回复: 我写的不是MFC



夏大伟

2018-06-09



是调用windows api中获取当前屏幕窗口尺寸的方法获得吗?

展开 ▾



风华神使

2018-06-09



楼上, cxx默认不用图形库, 用winapi

展开 ▾



立春

2018-06-09

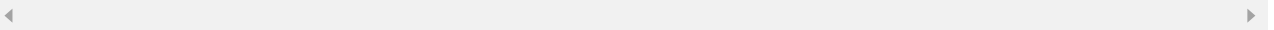


C++用的是MFC了.....

展开 ▾

作者回复:

mfc 全称 microsoft foundation classes, 微软基础库, 建立窗体可以使用 mfc, 也可以使用 ATL等等, 但基础代码还是windows api



呵呵

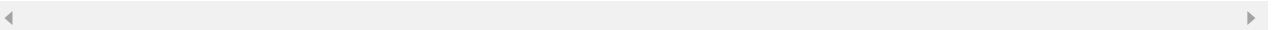
2018-06-09



用c++建立窗体, 默认用的什么图形库?

展开 ▾

作者回复: windows api





下载APP

