项目[desktopPet1\_Front]源码帮助文档

注：部分次要内容会不介绍。

这里的一些介绍会结合桌宠管理器的情况来写。

介绍

桌宠前置模组。整体的结构为，一个扩展([ExpansionPet](#类_宠物扩展))对应一个宠物，其中一个宠物可以由多个部件([PetObject](#类_宠物对象))组成，并由扩展管理这些部件的行为，这些部件就是一个控件。

桌宠前置和桌宠管理器基本上只提供了透明窗口的界面显示，如果要写桌宠基本上等于从零开始写，那也就是说基本没啥限制要写啥都行。

如何制作一个桌宠

为你的模组入口类继承[ExpansionPet](#类_宠物扩展)，并实现里面的方法即可，然后通过传入的桌宠管理器的接口调用[AddControl](#接口_桌宠管理器接口_AddControl)，就能向界面添加控件。如果需要一个模组有多个桌宠的话则继承[IExpansions](#接口_多宠物扩展)，并实现里面的方法即可。

前置模组内容

[desktopPet1\_Front.Expansion.Apis.IExpansions//用于多个宠物扩展的接口 2](#_Toc7053)

[desktopPet1\_Front.Expansion.ExpansionPet//宠物扩展 2](#_Toc26915)

[desktopPet1\_Front.Pet.PetObject//宠物对象、部件 3](#_Toc1788)

[desktopPet1\_Front.PetManager.Apis.IPetManager//桌宠管理器接口 4](#_Toc10452)

[desktopPet1\_Front.PublicClass1.IO.Get1//方便获取图片文件 4](#_Toc25492)

[desktopPet1\_Front.PublicClass1.PerRightMenu//右键菜单 5](#_Toc23333)

|  |  |
| --- | --- |
| desktopPet1\_Front.Expansion.Apis.IExpansions//用于多个宠物扩展的接口 | |
| 介绍 | 用于多个宠物扩展的接口 |
|  | List<[ExpansionPet](#类_宠物扩展)> GetExpansionPets()  获取扩展。当入口类继承该接口时，该方法会被桌宠管理器调用，并将获取的每个扩展添加到管理窗口上 |
| desktopPet1\_Front.Expansion.ExpansionPet//宠物扩展 | |
| 介绍 | 宠物扩展，建议重写的时候使用override |
|  | string Title  扩展的标题。当鼠标在桌宠管理器的该扩展的添加桌宠按钮(这部分是桌宠管理器模组的部分)上方时会显示该文本 |
| PetManager.Apis.IPetManager petManager  桌宠管理器的接口。可以通过该接口添加或删除部件(控件) |
| event Action Loaded  扩展加载完成时触发。由扩展触发。没啥用 |
| virtual void PetManagerLoaded(PetManager.Apis.IPetManager petManager)  桌宠管理器加载完成后会调用。可以在这里将传入的接口赋值给[petManager](#类_宠物扩展_petManager) |
| virtual void AddPat()  用于实现如何添加一个扩展宠物。该方法会被多次调用。当用户点击管理器中添加你这个扩展宠物的按钮时会调用该方法 |
| virtual void DelPat(Pet.PetObject po)  用于实现如何将一个扩展宠物删除 |
| virtual void ClearPet()  用于实现如何将全部扩展宠物删除。当用户点击管理器中的清空按钮时会调用该方法 |
| desktopPet1\_Front.Pet.PetObject//宠物对象、部件 | |
| 介绍 | 宠物对象、部件。这是一个控件，默认外观为宽高50、背景色白色、居右、居顶 |
|  | PetObject MainPetObject  用于存放主部件，当该部件是它的子部件时。该变量默认为null |
| List<PetObject> ChildrenPetObject  用于存放子部件，当该部件是它们的主部件时。该变量默认为null |
| Action<PetObject> Move  当调用[SetX](#类_宠物对象_SetX)或[SetY](#类_宠物对象_SetY)时触发，或手动调用OnMove方法触发 |
| Action<PetObject> RightClick  当右键点击部件时触发，或手动调用OnRightClick方法触发 |
| Action<PetObject> LeftClick  当左键点击部件时触发，或手动调用OnLeftClick方法触发 |
| virtual void SetImg(ImageSource img)  用来方便设置部件的图片之类的吧 |
| virtual void SetX(double v)  virtual void SetY(double v)  设置部件XY坐标。这两个方法已经实现了(通过设置控件的Margin属性实现修改位置)，重写也可以 |
| virtual double GetX()  virtual double GetY()  获取部件的XY坐标 |
| desktopPet1\_Front.PetManager.Apis.IPetManager//桌宠管理器接口 | |
| 介绍 | 桌宠管理器接口 |
|  | event Action Timer  计时器。在桌宠管理器中会以每秒30次的频率触发，建议不要用这个原本甚至打算不启动计时器的，主要这东西是不清楚要用来干嘛但还是做出来了，不知道用了会有什么问题 |
|  | event Action<System.Windows.Input.KeyEventArgs> KeyDown  在窗口上按下按键时触发 |
|  | event Action<System.Windows.Input.KeyEventArgs> KeyUp  在窗口上松开按键时触发 |
|  | void AddControl(Pet.PetObject po)  添加控件 |
|  | void DelControl(Pet.PetObject po)  删除控件 |
| desktopPet1\_Front.PublicClass1.IO.Get1//方便获取图片文件 | |
| 介绍 | :p |
|  | BitmapImage GetBitmapImage(string path)  path：文件的完整路径  获取图片文件的BitmapImage。会抛异常，记得接住了。直接把返回值赋给ImageSource就行了 |
| desktopPet1\_Front.PublicClass1.PerRightMenu//右键菜单 | |
| 介绍 | 有样式的右键菜单。好像使用的是这个前置模组里的样式文件，其实不用这个类直接使用ContextMenu应该也可以，因为主程序里有添加了ContextMenu的样式，当时可能是为了防止之后主程序换成其它的项目导致没有ContextMenu样式才又搞了这个，其它的模组项目好像都有好几份相同的样式文件可能都是这个原因 |