基于 CEGUI 的界面框架设计与实现①

黄伟强[1]　夏科艺[2]　(1 华南师范大学 网络中心 广东 广州 510631;
2 广州博冠信息科技有限公司 广东 广州 510665)

摘　要： 通过对界面开发库 CEGUI(Crazy Eddie's GUI)的研究基础上，采用归纳与总结、流程优化及代码与框架重构的思想实现对 CEGUI 界面库的 python 脚本封装，形成框架而进行界面开发，并以此框架开发界面编辑器，实现对界面开发流程优化，降低界面开发成本以及提高界面开发效率。通过游戏界面开发框架脚本化，对产品开发起到良好的促进作用。

关键词： 界面开发 封装 框架 CEGUI python 编辑器

Design and Implementation of User Interface Framework Based on CEGUI

HUANG Wei-Qiang[1], XIA Ke-Yi[2]
(1.Center of Network,South China Normal University, Guangzhou 510631, China;
2.Boguan Tenglong Information Tech Ltd, Guangzhou 510665, China)

Abstract: Through applying the CEGUI(Crazy Eddie's GUI) user interface LIB research, this paper accomplishes the CEGUI's python encapsulation by using induction, flow optimization and frame Refactoring. User interface development and User interface editor can use this CEGUI python frame to build, optimize the developing flow, reduce the cost of User interface development and improve efficiency.

Keywords: UI development; encapsulation; frame; CEGUI; python; UI editor

1　引言

目前游戏业已经形成一个发展快速的产业链，游戏产品层出不穷，市场需求巨大，而作为游戏产品重要的界面开发往往对游戏产品质量起到重要作用。因此对游戏产品的界面开发做研究，如何提高界面开发的效率，降低成本，提高可维护性和对市场的适应性是很有必要的。

CEGUI(Crazy Eddie's GUI http://www.ceg-ui.org.uk)[1]是一个基于LGPL协议,完全面向对象设计，开放源代码的游戏界面开发库。CEGUI 开发者的目的是希望能够让游戏开发人员从繁琐的 GUI 实现细节中抽身出来，以便有更多的开发时间可以放在游戏性上。

CEGUI 给游戏开发者一套公共的开源界面框架，并且稳定高效，开发者都可以在其上面根据自己的需要扩展，开发，制造出更适合自己项目的游戏界面框架。本文对游戏界面框架进行整体设计，设计实现框架结构的各个子系统，建立一个基于 CEGUI 框架为基础，以 python[2-4]脚本封装的界面框架。

2　游戏界面框架整体设计

2.1 基于 CEGUI 的游戏 UI 框架的设计原则

通过界面开发的经验，总结出一些界面开发总应用上的共性的东西，然后进行整合，形成一个更高层次的东西，这就是 UI 框架[5]。框架源于应用，却又高于应用。将框架设计成尽量于应用的偶合度降低，是框架设计的目标。界面框架是界面设计领域的框架，只负责界面的应用设计，本文所指的框架基于 CEGUI 为基础的框架二次封装，因此这个框架是和 CEGUI 的

① 收稿时间:2009-03-04

开发应用紧密偶合在一起的。根据对 CEGUI 的原理及应用研究，在归纳 CEGUI 的优缺点后，制定基于 CEGUI 的游戏 UI 框架的设计原则：

(1) 框架应该要保持 CEGUI 原有的窗口组织中父子窗口树状关系

树状的窗口组织关系，是由 CEGUI 的渲染方式决定的。CEGUI 只提供从根窗口执行渲染所有窗口绘制的方式，因此将各窗口组织成一颗树，才能保持其基本绘制特性。

(2) 控件窗口全局唯一性在应用时应该避免

默认情况下 CEGUI 内部生成窗口会自动分配一个单一名字，如“__cewin_uid_68”，而如果给定了窗口名字，如果 CEGUI 已经生成过同样名字的窗口，则创建窗口就会出错。程序编码时，如果动态创建窗口，则可能因为不知道已经存在窗口而创建窗口异常，很多时候，创建窗口就是希望即创即用，避免窗口名字全局唯一性，可以解决这个问题。

(3) 框架应该更方便的创建控件窗口

对于控件的大小，定位布局等应该更加方便。这样对程序员的编码工作可以节省大量重复劳动，提高程序员编码效率，减少代码量。

(4) 封装的窗口应该对消息事件机制做改善，尽量使界面逻辑和显示分开

CEGUI 提供的消息机制只针对每一个窗口进行注册回调处理，这样的设计可能会令程序员将逻辑处理都放在每一个窗口内部去做，造成逻辑与显示分离不开。

(5) 封装后的框架应该提供对应的窗口序列化与反序列化工作

这样便于制作界面编辑器，达到工具化设计、开发界面，给开发工作效率带来更好的影响力

(6) 框架应该维持 CEGUI 的属性操作便利性

(7) 封装后的框架应该提供对应的窗口序列化与反序列化工作

这样便于制作界面编辑器，达到工具化设计、开发界面，给开发工作效率带来更好的影响力。

(8) 封装后框架要提供新界面编辑器

界面编辑器生成的序列化数据，应该很方便于界面的生成与维护。

2.2 明确框架基本结构，进行结构子系统分析

框架总体结构主要包含：窗口控件子系统，消息驱动子系统，序列化子系统，工具子系统。框架结构如图 1：

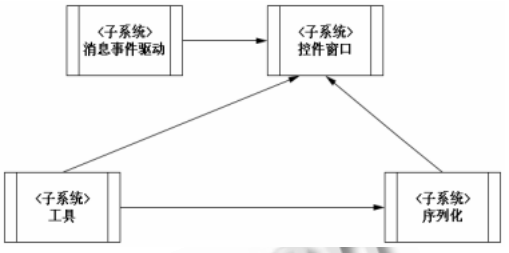

图 1 框架结构

界面框架的核心组成是控件窗口子系统，通过对控件窗口子系统的组织、管理、驱动等操作，使得整个框架能够满足应用需求。框架的其他部分主要因控件子系统的设计而产生和改进。消息驱动子系统是控件子系统内部消息流驱动的关键部分，其充当了框架在应用逻辑编码与设计时给予程序员一种合理的消息事件编程方式，通过消息驱动子系统的工作，可以让框架中各松散的组件有机的联合起来，实现更为复杂的逻辑界面。工具子系统承担了框架中经常性的方法、通用模块的包装，提供了必要的函数、类、模块等供框架解决各种经常性需求。序列化子系统则是专门针对控件窗口组件的序列化和反序列化操作，以完成窗口的序列化存储。

3 框架子系统的设计与实现

3.1 控件窗口子系统设计

控件窗口子系统在组织结构上由窗口基类和派生的子控件组成，通过对窗口基类及派生子控件类的设计，实现窗口生成流程，参数传递，窗口名字唯一性冲突处理，窗口属性等逻辑上的设计处理。控件窗口的设计，根据 CEGUI 的窗口特性，也需要保持基类窗口及子类化控件窗口的方式组织。

3.1.1 首先进行窗口基类设计 CWinBase

作为窗口类基类考虑几点规则：

① 因为基类不知道派生类如何设计参数，基类可以适应多变参数方式进行初始化。

② 基类需要固定初始化流程，派生类只关心需要关心的初始化步骤。

运用 python 封装脚本如下：

```
class CWinBase(object):
```

```
#定义默认的初始化函数，可接收不定参数和字典参数，采用延时创建对象方法
def __init__(self,*args,**kwargs):
#创建对象前准备工作函数
    self.preCreate()
#创建对象时调用的函数
    self.create(*args,**kwargs)
    try:
#创建时调用回调函数
        self.onCreate()
        except:
            self._clear()
#定义预创建函数接口
def preCreate(self):
    pass
#定义创建函数通知接口
def onCreate(self):
    pass
#定义创建函数接口
def create(self,*args,**kwargs):
    raise Exception("必须重载 create 函数")
    def _clear(self):
    pass
```

3.1.2 其次设计具体控件窗口的继承的父类 CWin

窗口类 CWin 主要实现窗口的初始化构建，提供默认窗口类型创建，CWin 类提供了窗口的各种公共使用的方法、属性、消息传递等共性的东西。

运用 python 封装脚本如下：

```
class CWin(CWinBase):
    def
create(self,parent=None,pos=None,size=None,winid=0,name=None,wintype="DefaultWindow",winname=None): self.win=self._create_window (win-type, winname)
#创建窗口
    self._win = self.win
#保存窗口引用
    self.set_name(name)
#设置封装对象名
    self.set_parent(parent)
#设置父窗口
    self.set_winid(winid)
#设置窗口 ID
    self.set_pos(pos)
#设置窗口位置信息
    self.set_size(size)
#设置大小
```

主要是提供一致的 位置，大小，窗口 ID，名字，窗口类型等参数，使得框架在应用于代码编写时，可以一行代码即可生成具体的窗口，并且对全局名字唯一的依赖改成不依赖。

3.2 消息驱动子系统设计

消息驱动子系统设计上主要由两方面实现处理：

(1) 编写消息驱动函数，嵌入到控件窗口子系统内的窗口类里，结合一起进行消息事件分发。

(2) 定义消息事件编号，使得消息事件驱动子系统的消息具有可扩展性及便利性，方便框架编程。

CEGUI 的消息事件采用按需注册使用机制，但消息不会在窗口的父子甚至多重窗口里传递分发。实际工作中的情况是，一个界面往往由一个个有逻辑关系的多控件组成。假设一个界面由 10 个子控件组成，每个子控件都需要处理一个事件，那么必须在代码里每一个控件写一个注册函数，并执行一次注册。代码量巨大，而且造成逻辑代码分布到每一个控件里去，不利于逻辑代码集中处理。编写、测试、维护成本都会随着增加。因此窗口类的消息事件驱动设计，考虑将消息进行多窗口传递分发，并且可由程序员在期望的某窗口里进行消息集中处理。比如一个界面由一个面板做底版，上面放 10 个子控件，那么可以将消息处理逻辑集中写在底板控件上，这样的好处是逻辑代码集中，便于维护管理，而且根据这个思路可以实现界面和逻辑分离。

利用 CEGUI 的 eventargs 参数的特性，每个参数都提供了产生事件的窗口的这个特点，因此将需要传递分发的事件集中到某个固定函数进行分发处理，而不需要分发的则只发给控件自身。在分发时，重新自行定义一套标识消息的类型参数，在分发时同时传递下去，这样就可以根据消息类型进行逻辑上的集中编程，消息的定义很灵活，可以根据不同的控件预先设计好的消息类型进行定义，然后再注册回调函数后分发出去即可。消息的数据类型采用整数型定义。

整个消息传递的框架如图 2 所示：

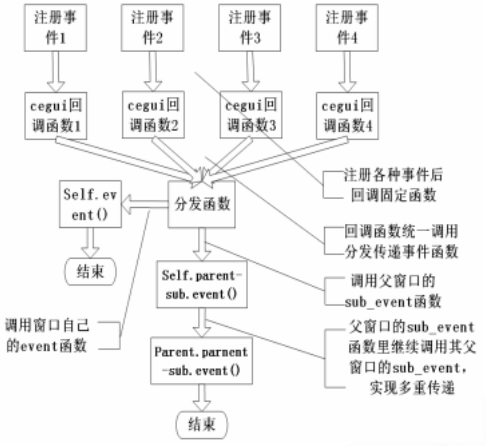

图 2 消息传递框架

通过对消息驱动的包装，似的框架的窗口组织可以将逻辑和显示最大程度的分开，并且将逻辑代码尽量聚合在一个地方集中编写处理。

3.3 序列化子系统设计

框架里窗口的序列化是根据窗口设计的特点来设计的。注意到窗口设计时，create 提供的参数包含了 parent,size,pos,image,name,winid,text 等参数，这些是默认参数。另外还隐含了窗口的类名，因此序列化时需要考虑将这些基本的属性进行序列化。

一个典型序列化数据设计为：

```
data=    {
        'cls':'CStaticImage',
        'name':'bgimage',
        'size':'{{0,334},{0,438}}',
        'pos':'{{0,206},{0,38}}',
'image':'imagefiles/char/image_shu.tga',
        'text':'',
        'winid':0,
        'childs':{
            'rename':{
                'cls':'CButton',
                'name':'rename',
                'size':'{{0,32},{0,19}}',
                'pos':'{{0,288},{0,213}}',
                'image':'char/btn_rename',
                'text':'',
                'winid':0,
                'childs':{
                },
            },
    }
```

3.4 界面编辑器设计

编辑器主要工作是实现界面元素排版，并生成序列化后的界面数据，以便程序动态利用序列化数据生成界面。编辑器的实现是依据框架基础实现的，使用的是框架里的控件进行设计。从图 3 编辑器的结构图里可知，编辑器由编辑处理、控件树管理、属性管理、控件资源管理、操作管理等 5 部分组成。

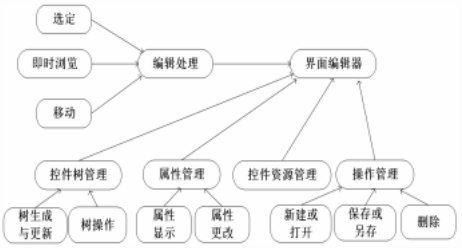

图 3 界面编辑器结构

4 结语

CEGUI 由于其开放的特性，随着越来越多的项目开发人员使用，将会不断得到完善，这也促进了这套界面库的良性发展。由于 CEGUI 的通用性，本文基于 CEGUI 界面库的二次开发形成的界面库也相对通用性更强，这对于降低界面开发成本具有现实意义。本文通过基于 CEGUI 的研究除了具有界面开发框架的理论方面提高的意义，还减少代码编写量，使策划和程序可分离设计，程序编写逻辑代码模块性更强，而且框架学习曲线不高，容易掌握，利于项目开发新成员的进驻，对游戏开发项目具有明显的价值产生。

参考文献

1 Cegui.org.uk. Crazy Eddies GUI System API. 2008. http://www.cegui.org.uk/

2 Lutz M. Programming Python, 2nd ed. 2001. 05.

3 Eckel B. Thinking in Python. 2002.

4 Lundh F. Python Standard Library. 2001.05.

5 周靖,张杰良,Richter J.框架设计:CLR Via C#.第 2 版,北京:清华大学出版社, 2006.