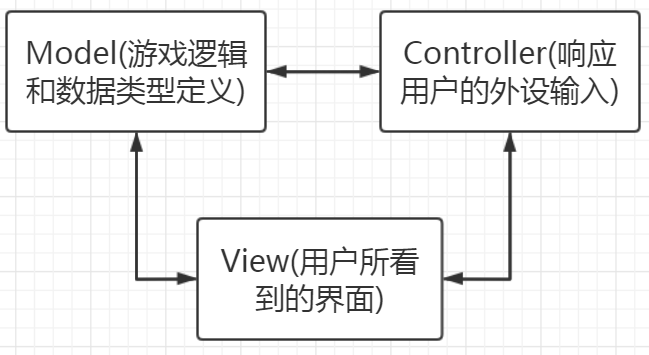
**PyGame Frame**

1. 简介:

PgFrame(PyGame Frame)是一个用于快速便捷的开发中小型2d游戏的python模块(不支持对大型游戏的优化)，结构上使用高度单元化的模块结构构建游戏（通过搭建MVC的形式完成游戏的制作），可以使程序员更多的将精力集中于游戏逻辑，提高开发效率。

程序结构上大致采用MVC的框架结构:



(各部分间的调用顺序)

Model: 算法部分 + 数据部分 (程序后台处理和数据管理)

View: 视图部分 (主要控制用户界面)

Contrller: 控制部分 (响应用户的外设输入)

PgFrame主要基于pygame(绝大部分功能依赖于此)和numpy，另外还需要下载pillow、pywin32(因此该模块**仅支持windows系统**)、pyinstaller(可以在安装pgframe后使用cmd命令pgframe support或者运行pgframe目录下的support.bat 来自动下载这些第三方库)

PgFrame在底层上继承了PyGame灵活快速的特点(这使得PgFrame能在运行速度上比Pyglet和Cocos2d快上不少)，在框架上比pyglet具有更经典实用的框架结构(MVC)，在高层上也具有高自动化的组件(Component)作为支持；PgFrame的缺点为资源占用较大(与相同的pygame游戏相比能达到数倍的内存占用与cpu占用(这取决于游戏的规模，当游戏规模适中时这个倍率将接近于1))，必须多个文件同时工作才能正常运行，平台仅局限于windows(使用了pywin32导致的，然而由于pywin32在程序段中占比低，后期可以通过兼容mac和linux的方式支持这些系统)，且不支持3d，更新较慢。

1. 入门使用

下载对应的python whl文件后pip安装。安装完毕后使用pgframe version查看pgframe版本，若成功显示版本则安装成功



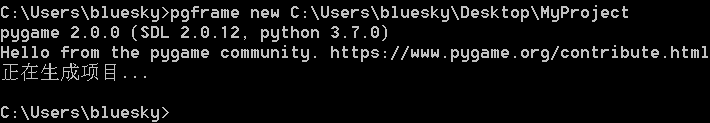
成功后若缺少开头提到的依赖库，使用cmd命令pgframe support自动安装这些库, 安装过程中没有报错即可完成安装.

接下来我们创建一个带有图片的hello，world!例程:

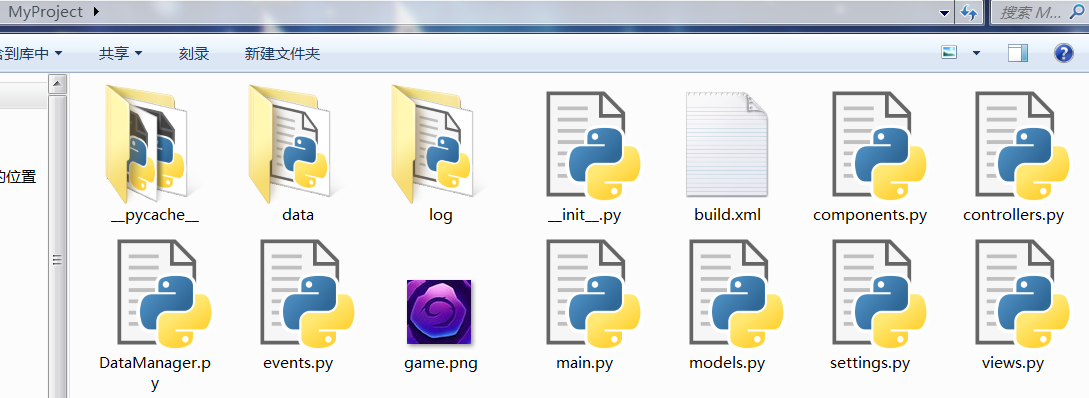
1. . 创建项目：

打开cmd，输入pgframe new [path]([path]是你选择的文件夹路径，项目将创建在这个路径下，该路径下已有的文件将被移除)

例如这里使用的路径是C:\Users\bluesky\Desktop\MyProject



打开对应的项目目录，里面应当出现如下文件:



**\_\_init\_\_.py** : 所有的\_\_init.py\_\_文件都没有实际的意义(这是在库安装时需要的)

**data目录**: 默认的资源文件存放的位置，具体路径可以在settings.py中更改

**log目录**: 默认的日志文件存放的位置，具体路径可以在settings.py中更改

**build.xml**文件: 打包配置文件

**view.py**：MVC的View部分，指用户看到并与之交互的界面。与传统的MVC有所不同的是：所有视图在程序中是真实发生了的，视图的主要功能是划分结构和层次，并作为载体组织其余的模块进行工作。

**controllers.py**：MVC的Controller部分，控制器接受用户的输入并调用模型和视图去完成用户的需求，与传统的MVC有所不同的是：控制器可以进行预判断而选择不同的算法和视图进行操作

**models.py:** MVC的Model部分，模型用于实现游戏逻辑和有关算法，拥有最多的处理任务。与传统的MVC有所不同的是：模型还具有定义某种数据库中的数据的结构的作用。

**components.py**: 组件，完成主要的显示工作以及各种各样的**几乎所有**的衍生功能，且可以在一个View中部署已有的任意数目的components来实现想要的功能。

要自定义component就需要定义对应的帧函数effect(对应于pygame中的while中的一次循环)。 使用内置的已经编写好了的component即可满足日常编写需求, 所以该模块大部分时间并不需要用户编写，节省了开发时间。

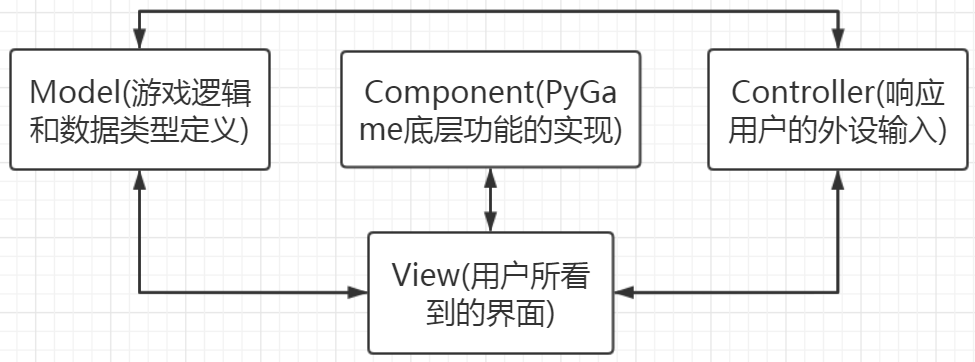
**event.py**: 事件管理，由于事件可以临时定义，所以该模块一般情况下用不到（以后可能会取消这个模块）

**settings.py**: 游戏项目的设置，比如窗口刷新率、窗口尺寸、窗口大小等等的设置

**main.py**: 项目运行入口，不用用户编写。完成其他文件的编写后执行main.py即可启动游戏。

**DataMaganer.py**: 数据库管理程序，这是一个单独的基于tkinter的数据库管理，需要读取models.py中定义好的数据结构

所以，更贴切一点的说， PgFrame的结构应当为MVC + C的结构，记做MCV-C:



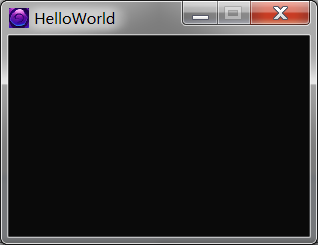
**一般来说，只需要用户先定义好settings.py，然后把资源文件放置于对应的文件夹下，最后编写models.py、controllers.py和views.py即可**

1. .编写项目:

①.按照第一点中的编写顺序，我们先修改settings.py中的内容:

修改窗口大小为(300，200), 窗口标题为HelloWorld

修改完成后运行main.py查看效果:



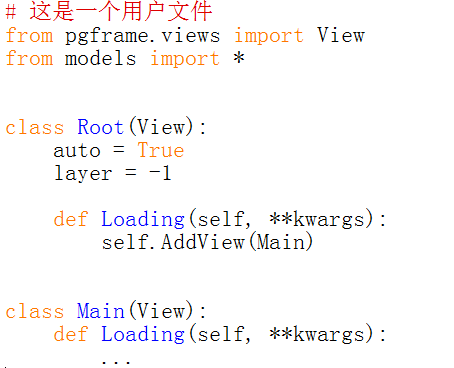
②.然后我们选择一张300x200的png格式的图片(test.png)放置于data/img/下:



③.最后编写文件:

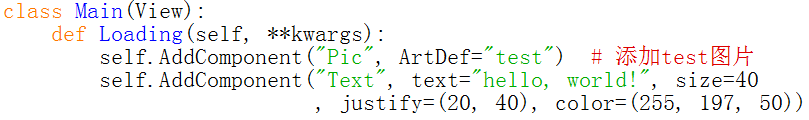
由于我们的目的只是显示hello,world!和图片，所以这里我们只需编写views.py即可.

打开views.py文件，里面已有一段内容:

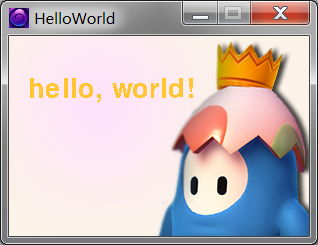


我们在Main.Loading下添加两行代码，分别实现图片和文本的效果:

self.AddComponent("Pic", ArtDef="test") # 添加test图片

self.AddComponent("Text",text="hello,world!",size=40,justify=(20,40),color=(255,197,50))

还是运行main.py查看效果



(至此我们的入门项目就算完成了，更多的信息请查看函数文档和具体的实例)

1. .\*打包项目:

若依照安装步骤安装了pyinstaller，这可以对游戏项目进行打包，打开build.xml，修改好有关内容后关闭:



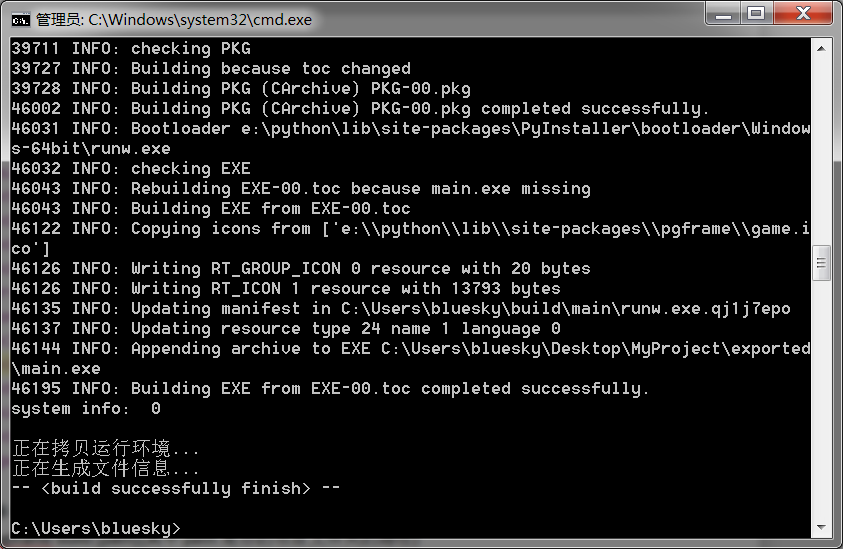
(这里作为单个文件打包)

Cmd命令pgframe build [path](其中path是你的项目文件夹的路径)

在该情形下，这里应当输入:

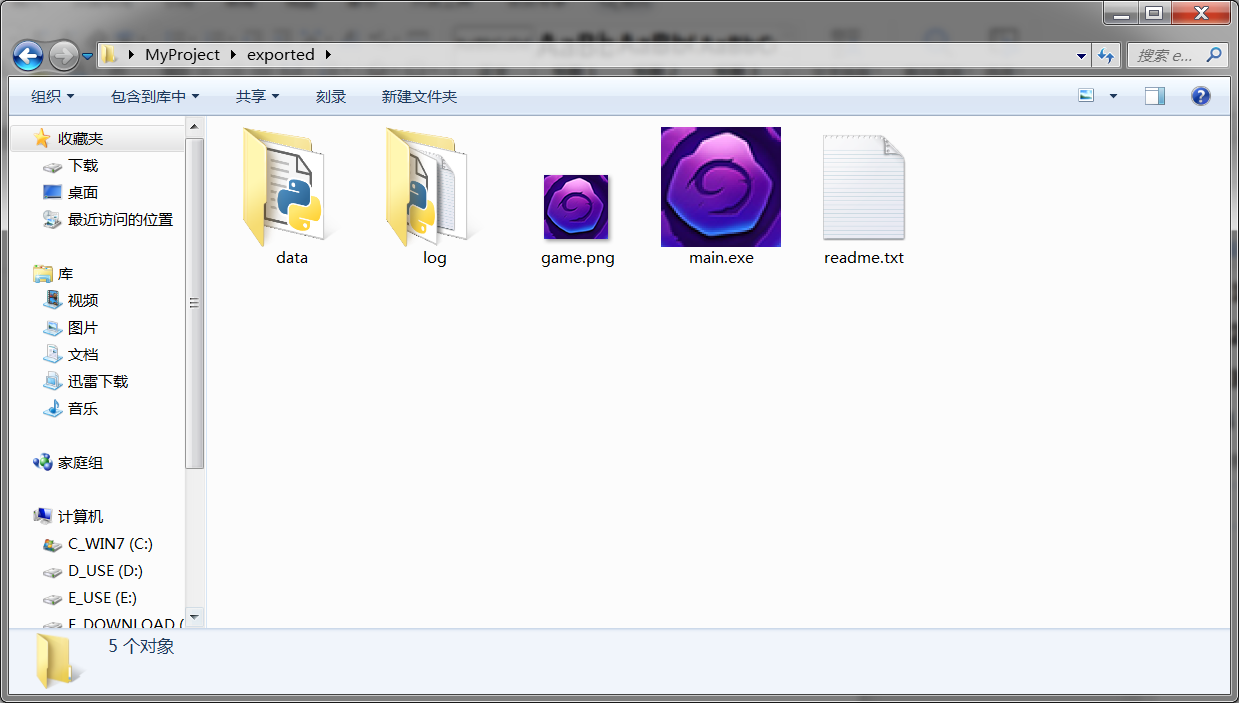
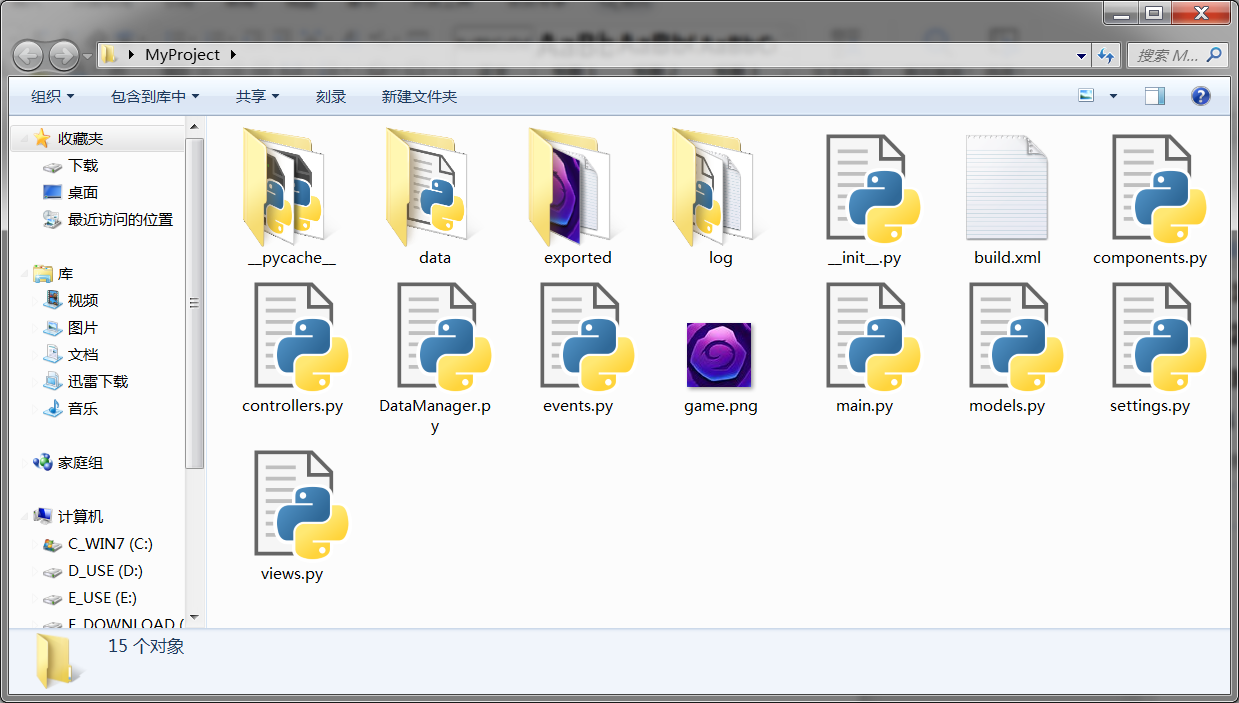
pgframe build C:\Users\bluesky\Desktop\MyProject

回车后，若没有出现异常，则应当出现如下的画面:



（最容易出现的问题则是.dll文件无法拷贝的情形，请关闭杀毒软件后重试）

此时在项目文件夹下应当出现一个exported文件夹，里面就是我们打包好的.exe文件了:



（有时可能会出现游戏所需资源没有拷贝完全的情况，请用户手动拷贝即可）