自定义宠物

宠物形象文件夹的结构

我们的宠物形象的配置，只需要根据模板改写一个配置文件（JS格式）和形象需要的贴图、背景、形象等图片即可。配置文件决定宠物的行为的文件。
比如，在我们提供的 demo 中，只有四分文件，其中只有`conf.js`是决定宠物的文件。demo 中的 SVG 文件仅为宠物形象图。

导入宠物形象

欲导入外来的宠物形象，请遵循以下步骤：

一、确保自己获得的宠物形象是有效的宠物形象
二、将宠物形象的**文件夹**放置到`【项目根目录】/assets/pet/`目录中。注意不是将宠物形象文件夹内容直接放置到该目录中。
然后将文件夹重命名为自己想要的内容。（注意：为了防止文件编码导致配置读取异常，我们不建议你用中文命名。）
三、修改程序的配置文件`【项目根目录】/assets/conf.js`，
- 找到`petroot`字段，将后面引号内的内容改为形象的宠物形象的**文件夹**名称。
- 找到`petconf`字段，将后面引号内的内容改为形象的配置文件名称。

四、重新启动宠物程序或重新加载宠物，检验效果。

如：

```
conf={
  // ……
    petroot:"【demo】",
  petconf:"〖conf.js〗",
  // ……
}
```

本举例中，程序配置会引导程序根据`【项目根目录】/assets/pet/【demo】/〖conf.js〗`来生成宠物形象。

简单自定义宠物形象（pet/.../__.js）

本内容需要 JavaScript 编程基础。
我们默认提供一个 DEMO 宠物形象。该形象的文件存储于`【项目根目录】/assets/pet/demo/`文件夹中。
宠物形象的配置文件主要有以下成分

成分	类型	用途
`pet`	Object	定义宠物和主人的信息、定义宠物形象。
`dict`	Object	定义点击宠物时宠物会回应的互动消息内容。
`dict_eating`	Object	定义给宠物喂食时宠物会回应的互动消息内容。
`dict_shower`	Object	定义给宠物洗澡时宠物会回应的互动消息内容。
`config_petitems`	Object	列举在宠物界面右上角显示的物品。
`config_petmenu`	Object	列举在宠物界面右键菜单中显示的内容。
末尾JS代码段	JS代码	宠物配置文件加载完成后，调起预加载。**不建议非专业人员修改。**

dict、dict_eating、dict_shower 配置

这三个部分定义与宠物时宠物会回应的互动消息内容。

这三个部分均通用一套模板。我们此处以dict为例：
列表两侧用方括号包围，列表一项可以接受`text`和`Object`两种类型的数据。
实际使用时，函数`popup(……)`会根据列表一项的内容类型自动作出反馈：

- text：互动反馈时，跳出的对话框内容即为文字内容，按钮显示为“关闭”字样。
- object：互动反馈是，跳出的对话框内容为`content`的内容，按钮为`button`内规定的内容。*后期可能更新，支持多个按钮。*

注意：除最后一项外，每一项的末尾都需要用半角逗号`,`进行分隔。

比如：

```
dict=[
    "你好",
    {content:"很高兴认识你",button:"我也是！"}
  {content:"要保持微笑呀～",button:"一定",img:pet.imgs.smile}
];
```

此段代码中，与宠物一次互动时，程序会在下列三个个操作中三选一弹出：

- 弹出一个带有“你好”二字和“关闭”按钮的对话框。
- 弹出一个带有“很高兴认识你”和“我也是！”关闭按钮的对话框。
- 弹出一个带有“要保持微笑呀～”和“一定”关闭按钮的对话框，同时宠物形象改为在`pet`中早已设定的`smile`对应图像。

config_petmenu 配置

方法一

`config_petmenu`定义宠物右键菜单内容。列表两侧用方括号包围，列表项要求类型为`object`。
列表一项须要含有`label`和`exec`的内容。`label`用于说明菜单项内容，`exec`表示点击后的操作内容（JS代码）。

```
config_petmenu=[
  // ……
  {
    label:"宠物信息",
    exec:function(){  // 在这里放下需要执行的内容
      pet_info();petmenu_close();
    }
  },
  // ……
]
```

如此例子，这是`config_petmenu`中的其中一项。这项规定了显示“宠物信息”、点击后执行函数`pet_info`（打开宠物信息）和`petmenu_close`（关闭菜单）的菜单项。

方法二

利用`petmenu_add(config)`函数加入内容。config 是菜单一项的 Obejct。
届时本例子内容将应当被改成：

```
petmenu_add(
  {
    label:"宠物信息",
    exec:function(){  // 在这里放下需要执行的内容
      pet_info();petmenu_close();
    }
  }
);
```

这种方式只能每执行一次补充插入一项。

config_petitems 配置

有点类似于配置 config_petmenu 的方法。

方法一

```
config_petitems=[
  // ……
  {
    label:"蜘蛛",
    exec:function(){ // 这里放入需要执行的代码
      petload(pet.imgs.scary);facelocked=true;setTimeout("facelocked=false",conf.
特殊形象固定时长)
    },
    src:'spider.svg'
  },
  // ……
]
```

方法二

利用`petitem_add(item,drag_id)`函数加入内容。*`drag_id`字段其实可以忽略。*
届时本例子内容将应当被改成：

```
petitem_add(
  {
    label:"蜘蛛",
    exec:function(){ // 这里放入需要执行的代码
      petload(pet.imgs.scary);facelocked=true;setTimeout("facelocked=false",conf.
特殊形象固定时长)
    },
    src:'spider.svg'
  }
);
```

自定义程序配置（conf.js）

`【项目根目录】/assets/conf.js`的成分及其用途为：

成分	类型	用途
`petconf`、`petroot`	String	指定宠物形象配置文件
`plugins`	Object	控制插件
`stat`	Object	状态插件的配置
`popup_delay`	Int	调整互动消息显示时长

用户可以根据自己需求在本文件中加入自定义函数。

上文已经介绍过如何通过修改`conf.js`改变宠物配置文件，此处不再赘述。

选择插件

插件分为宠物界面插件和信息界面插件。宠物界面插件和信息界面插件不一定互通。在`conf.js`中，`plugins`部分负责记录宠物界面插件，`win_plugins`负责记录信息界面插件。

一、获取插件文件（文件夹或 JS 文件），并将其放入`plugins`文件夹内。

二、判断插件类型。带有后缀“_win.js”的是信息界面插件，直接“.js”结尾的则是宠物界面插件。

三、在列表中放入插件文件相对于`plugins`文件夹的位置。

- 宠物界面插件 无须“.js”后缀名
- 信息界面插件 无须“_win.js”后缀
- 每一层目录后方用`/`分隔。

如：

```
conf={
  // ……
    plugins:[ // 宠物界面插件
        "【甲】",
    ],
    win_plugins:[ // 信息界面的插件
        "〖乙〗",
    ],
  // ……
}
```

此配置要求下，宠物界面会加载`【项目根目录】/assets/plugins/【甲】.js`，信息界面会加载`【项目根目录】/assets/plugins/【乙】_win.js`

注意：部分插件可能会带有自己的资源，我们强烈建议这些插件将所有的资源放入自己的独立文件夹中。

状态插件的配置

根据中文提示修改配置。注意：延时的单位为毫秒。