BigEscape 项目说明

描述

BigEscape 是一款射击类游戏，类似于”雷电”游戏。由 cocos2d for c++ 引擎制作,下面请看游戏中几张截图,然后会对其中的每张图做简单的介绍。这里的一些图片资源是拷贝 MoonWarriors 的资源，感谢提供。虽然不是直接提供的。

图 1: 游戏开始页面

图 2：游戏选项页面

图 3：游戏关于页面

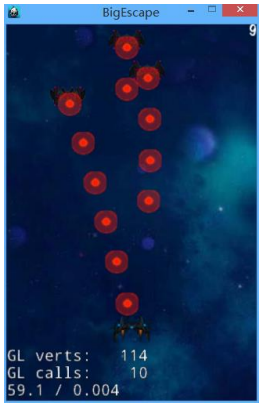


图 4 游戏界面

图 5：游戏结果

第一阶段： 建立 cocos2d 工程

既然使用 cocos2d 引擎开发游戏，说明你肯定对 cocos2d 有一定认知。但是，认知得总归要发展到编写程式的阶段，下面就简单介绍如何建立 cocos2d 引擎的 Helloworld 工程。这是关键的，因为后面的项目必须对这个阶段的内容了解。

工具: Python 2.7 、VS2012+、cocos2d 源码包。

这里注意 cocos2d 的源码包有很多版本，这里我们从官网(www.cocos2d-x.org)

下载最新的源码包解压。第一步:解压后的 cocos2d-x-3.x/build/目录，使用 VS2013 编译运行，测试是否能够编译通过，如何能够通过则进行第二步，不通过需要自己解决编译问题。第二步:安装工程环境。这步很关键因为后面的游戏项目就是需要在工程环境的前提下使用 cocos 指令来建立 VS2013 工程项目，搭建环境需要使用 Python2.7,使用 python 运行 cocos2d-x-3.x/setup.py 运行自动搭建即可。工作搭建好以后在命令控制台 cmd 下运行 cocos –h 指令检查是否安装完好。

下面是截取其中的用法说明

```
Available commands:
        run             Compiles & deploy project and then runs it on the target
        luacompile      minifies and/or compiles lua files
        deploy          Deploy a project to the target
        package         Manage package for cocos
        compile         Compiles the current project to binary
        framework       Manage frameworks for the project
        new             Creates a new project       建立新的 cocos2d 工程
        jscompile       minifies and/or compiles js files
Available arguments:
        -h, --help      Show this help information
        -v, --version   Show the version of this command tool
Example:
        F:\cocos2d-x-3.6\cocos2d-x-3.6\tools\cocos2d-console\bin\/cocos.py new --help
        F:\cocos2d-x-3.6\cocos2d-x-3.6\tools\cocos2d-console\bin\/cocos.py run --help
```

建立工程使用 cocos new 命令

```
C:\Users\CastroLiu>cocos new -h
usage: cocos new [-h] [-p PACKAGE_NAME] -l {cpp,lua,js} [-d DIRECTORY]
                 [-t TEMPLATE_NAME] [--ios-bundleid IOS_BUNDLEID]
                 [--mac-bundleid MAC_BUNDLEID] [-e ENGINE_PATH] [--portrait]
                 [--no-native]
                 [PROJECT_NAME]
Creates a new project
positional arguments:
  PROJECT_NAME              Set the project name
optional arguments:
  -h, --help                show this help message and exit
  -p PACKAGE_NAME, --package PACKAGE_NAME
                            Set a package name for project
  -l {cpp,lua,js}, --language {cpp,lua,js}
                            Major programming language you want to use, should be
                            [cpp | lua | js]
  -d DIRECTORY, --directory DIRECTORY
                            Set generate project directory for project
  -t TEMPLATE_NAME, --template TEMPLATE_NAME
                            Set the template name you want create from
  --ios-bundleid IOS_BUNDLEID
```

Set a bundle id for ios project

--mac-bundleid MAC_BUNDLEID

Set a bundle id for mac project

-e ENGINE_PATH, --engine-path ENGINE_PATH

Set the path of cocos2d-x/cocos2d-js engine

--portrait Set the project be portrait.

lua/js project arguments:

--no-native No native support.

上面的说明，常用的只有 cocos new 指令

BigEscape 的工程建立使用如下指令:

```
C:>> Cocos new BigEscape –l cpp –d F:/
```

上述指令代表的意思是: 使用 cocos2d 引擎建立 以 c++为编写语言，工程目录建立在 F:/盘，工程名为 BigEscape 的工程，当然也会自动建立相关目录。

原始的工程是简单的 HelloWorld，可以直接编译运行，然后我们的任务就是在这基础上添加个人项目内容。建立成功以后即可进入下一阶段。

第二阶段: 项目准备工作

这一阶段的任务是:搜集相关资源 包括 图片(png jpg etc.) 音效(mp3) 及动画整合包(plist and png)

工具: PS、TexturePacker 制作器

这里不说明说明见工程目录下资源文件树:

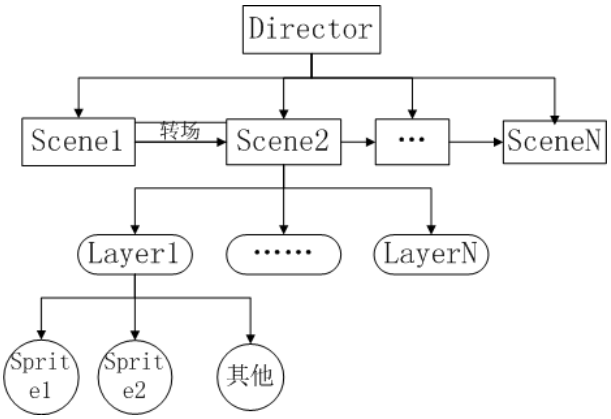
```
├──fonts
    arial.ttf 、Marker Felt.ttf 、Marker.ttf
├──Image
    1.png、2.png、3.png 、4.png 、5.png、6.png、7.png 、8.png、9.png、10.png、
    11.png 、12.png 、13.png、14.png、15.png 、16.png、17.png 、18.png、
    background.png、background1.png、background2.png、gameOver.png、header.jpg、
    loading.png、logo.png、musicOpt.png
│
├──Music
│        bgMusic.mp3、buttonEffet.mp3、explodeEffect.mp3、fireEffect.mp3、
mainMainMusic.mp3、shipDestroyEffect.m
│
├──Plist
│   ├──bullet ：bullet.plist 、bullet.png
│   ├──enemy：Enemy.plist、Enemy.png
│   ├──explosion：explosion.plist、explosion.png
│   ├──menu：menu.plist、menu.png
│   └──ship：ship.plist、ship.png
```

到此使用到那些资源

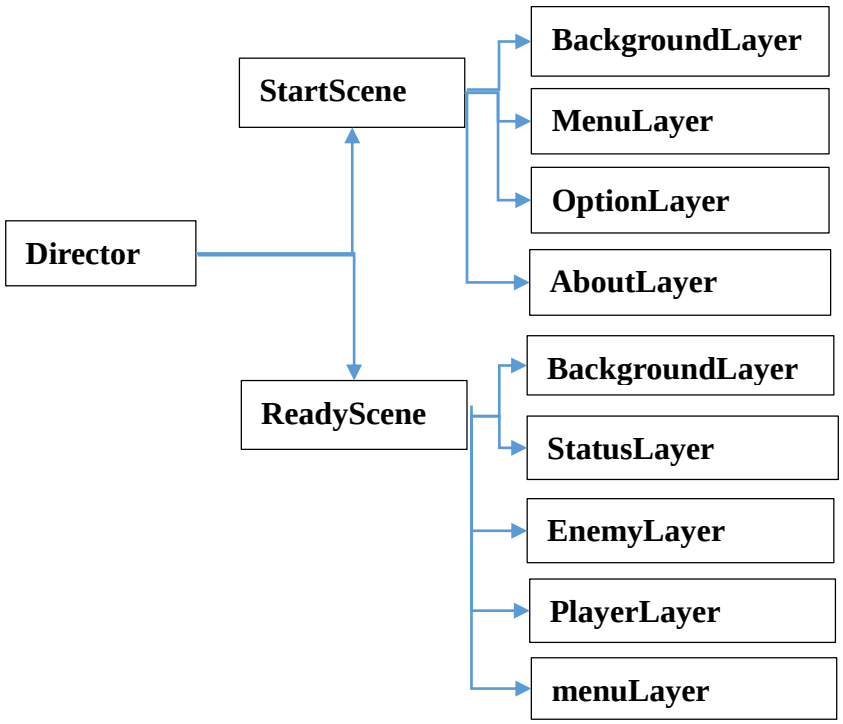
第三阶段：开始写代码

写代码时，为了追求更好的查看代码中的内容，代码的格式统一使用命令方法使用 函数名使用 **小驼峰** 命名法， 变量名使用 **匈牙利**命名法。另外对齐方式也要统一。

基本概念 ：具体知识请搜索文档 建议先去 http://www.cocos.com/doc/把基础知识点了解后再继续下一步的操作。下面是关于 cocos2d 引擎游戏管理框图。



现在框图已经有雏形，但是这只是总体设计方案，所以针对我们的游戏，画出如下的框图：



接下来就根据上图来编与代码了，现在如何编写程序又是问题。Cocos2d 引擎 C++ 对象编程，如何分离不同的模块功能，又是在考验我们。

首先是针对游戏的开始画面进行编程。

回到第 1 页，看图 1 并配合上图来思考。下面列举 开始场景的中 相应层 及元素：

背景层中有 闪烁的星星， 移动的飞船，大小变换的 logo；
菜单层中有 三个 中间对齐的 三个按钮；
选择层有 背景图片，三个开关按钮；
关于层：程序人员的相关信息；
下面则是游戏场景中的相关层及元素
背景层： 时刻变换的背景；
玩家层： 唯一的可以未得到结果前 移动 飞船， 它可以 发射子弹；
敌人层： 随机产生的敌人 伴随 它们所 持有的 武器 沿着非直接的 轨迹飞行并始终把目标锁定在 玩家身上；
菜单层： 按下空格后游戏暂停， 有两个菜单选项出现；
状态层： 活跃的“GameOver”，玩家的分数，一个返回菜单项。

游戏背景的移动思考：

在游戏场景中如何让飞机感觉自己在移动。立刻想到的是 背景 的移动，但是如何移动呢？最后想出的解决方案是使用两张 能够融合恰当的 连贯图，在 cocos2d 里 两张图片都为精灵(Sprite) ，然后就是精灵的 动作 及位置 的配合以达到背景在 连续移动的 场景

使用资源为以下两张 经过 PS 的版本：


然后就是如何使用两张图片有次序的流动。下面是让图片流动起来的关键语句

```
auto sprite1 = Sprite::create("Image/background2.png");
sprite1->setPosition(visableSize.width, 0);
sprite1->setAnchorPoint(Vec2(1, 0));

auto sprite2 = Sprite::create("Image/background1.png");
sprite2->setPosition(visableSize.width, sprite1->getContentSize().height);
sprite2->setAnchorPoint(Vec2(1, 0));

auto action1 = MoveBy::create(20, Vec2(0, 0-sprite1->getContentSize().height));
auto action2 = MoveTo::create(0, Vec2(sprite1->getContentSize().width,
sprite1->getContentSize().height));

sprite1->runAction(RepeatForever::create(Sequence::create(action1, action2, action1->clone(),
nullptr)));
sprite2->runAction(RepeatForever::create(Sequence::create(action1->clone(), action1->clone(),
action2->clone(), nullptr)));
```

关键是位置摆放及 动作时间的安排 。最终效果还凑合。

当前问题： 不能正常播放节点动画。
原因：

没有正确 重写 onEnter () 函数，在重写 onEnter（） 里 需要再次调用 Layer::onEnter() ,代码如下：ds

```
class startSceneLayer : public cocos2d::Layer
{
public:
    virtual bool init(void) override;
    virtual void onEnter(void) override;
    virtual void onExit(void) override;
    ……
}

void startSceneLayer::onEnter()
{
    Layer::onEnter();
    AudioEngine::resume(m_bgmID);
    this->resumeSchedulerAndActions();
}
```

音效的控制思考:

在写背景层代码时，考虑到了音效问题，具体点是如何有控制音效的播放、停止、暂停、恢复问题。因为这些控制是全局的：背景，击打音效。在基于上面的考虑以后,在实现操作方面： 已经有 newAudioEnging 类， 但是在场景中需要切换背景音乐中实现繁琐 且耦合性 太高，所以引入 类 musicControl，这个类帮助管理音效的播放。 此类使用是 **单例模式** ， 因为在整个游戏当中有且只有一个音效引擎。且使用开关位来控制不同的效果是否播放。这个方式很好正好将其它层分离开来，效果非常不错。

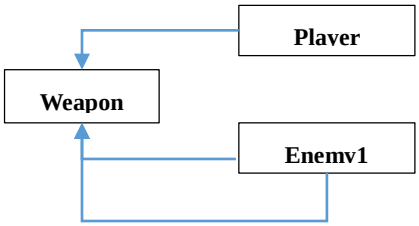
玩家射击思考

现在进入 玩家 的飞船控件模块编写，关于触摸这一块的编写有几个重要的内容是事件监听的添加 有 EnterBegan() Ended() Moved() Canceled() 要注意 OnBegan 是返回 bool 类型值的。为了简便游戏设计， 操作只能是用手来拖动玩家飞船类进行操作， 且玩家飞船为 不停地间隔发射子弹。

当然这里的子弹并不是一直在生成的 要考虑的是并不是一直在产生子弹，这里使用的技术是使用 **缓冲池** (BufferPool) 技术，这保证了 不用产生很多中间过程时间。 要考虑的是如何有间隔的发送 子弹 并从池中删除， 并在何时并回收子弹。 由谁来回收。 处理办法也是使用缓冲池来保存已经发送的 子弹，在玩家层 或者 敌人层来遍历 发射缓冲池， 并获取相应坐标，如果 子弹飞出 可见视图处将回收子弹。注意 cocos2d 是模仿 c# 里的内存管理机制，在 delete 析构其中的 Sprite 时会出现错误，所以编程中要考虑 retain() autorelease() release() 接口方法(不可重写) 继承于 CCNode 类, 使用 new 时就使用 retain() 来保存。析构里来 release() 对象。

为此，设计了武器类，此类的功能是让某个图层拥有武器，注意这里提到的是图层(layer)，因为控制子弹发射是图层的定时功能所实现。下面是武器类的此项目内的应用框图

武器类 Weapon



这里 player 只拥有一把武器，所以只有一条线连接，而敌人有很多，所以使用两条线来连接。下面介绍武器类的实现原理。

这里再次使用 **缓冲池** 的概念。在 Weapon 里创建两个类，分别为

```
cocos2d::CCArray * m_aShootedBullet;        //   子弹  缓冲池
cocos2d::CCArray    * m_aBullet;                //  子弹  缓冲池
```

分别是 已经射出去的子弹池 及 缓冲池内剩余的子弹池。技术解决是由于创建过程所消费的时间问题。

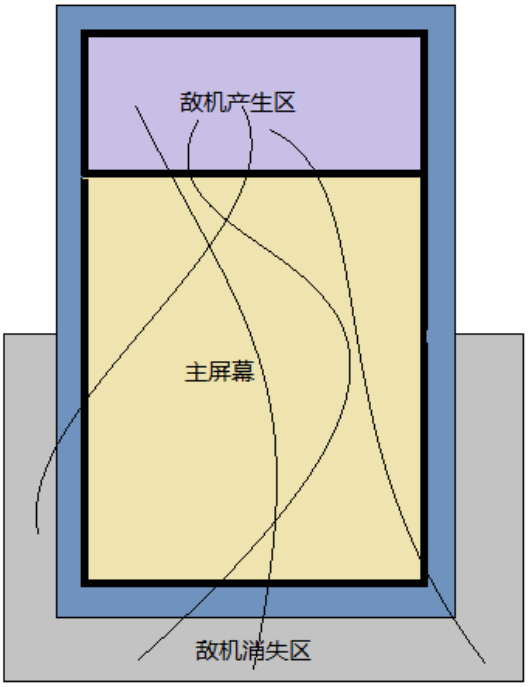
玩家武器的发射思考：

此工程是使用 点击 触发发射子弹功能来实现的，具体考虑是 Player(玩家)是武器的持有者，是否发射由持有者来决定，并不是由武器本身来决定，所有在 PlyerLayer 中添加了 (m_shoot)是否射击的属性。

编写过程中的技巧描述如下： 在处理触摸注册函数是使用 Lambda 表达式(**匿名函数**)来解决的，但是射击是需要获取对象本身的属性的,它使用到了[=](){}; 形式来加入引用到匿名函数里，这里引用 this 指针就相当于完成相当大的任务。

敌人的产生及飞行轨迹思考：

关键部分到了：敌机的原理区域图如下图所示，下面将介绍这些区域意义



敌机产生区：顾名思义敌机的生产位置

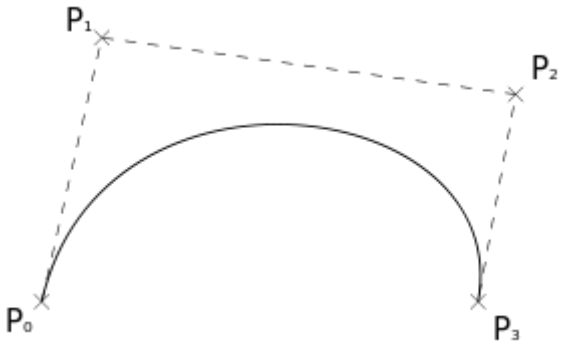
主屏幕　　：玩家所能见的可视范围

敌机消失区：敌机在这个范围内将自动回收(缓冲池机制)

那么上述那些细线当然就代表敌人的飞行轨迹了，如何产生这些轨迹呢？第一反应产生弯弯的曲线当然得是贝赛尔曲线上场了。

上图是解释其原理，敌机生成只能在上图中的区域生成，在最外层的方框图是敌机飞船的 最多可能飞去的区域当飞船飞去以后则可以回收敌机飞船并等待下次敌机产生。

第一步是随机产生回收点，即最终飞行的终点，而且这个终点必须是在回收范围内(敌机消失区)，而控件飞行曲线的弯曲程度则是设置如下图所示的两个控制点所得到的。



其中 p0 开始点, p3 结束点，p1 p2 贝赛尔曲线的控制点.P0 在图中“**敌机产生区**”随机产生，P3 在图中“**敌机结束区**”产生，P1 P2 本工程是使用 P0 与 P3 的 1/4 处 3/4 处偏移产生，在敌人类 class EnemyLayer 中可以查看验证，亦可修改添加其它动作加入。轨迹设置好以后就可以当敌机加入到已使用敌机缓冲池内直到回到，当然回收已经敌机的行为是肯定会发生的。敌机所持有武器也是在产生一个敌人时就已经绑定了。

让敌人的武器找准谁的思考：

观察者模式：观察者模式定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态发生变化时，会通知所有观察者对象，使它们能够自动更新自己。这句话正好可以用在这个问题上面。 敌人有很多个，而玩家只有一个。现在就是如何在代码上现实了。

当然玩家的射击伴随行为: 有相应的 **动画** 及 **音效** 在发生。这样才会有炫酷的感觉，当然真实一点是操作者点击时才会射击，这时才伴随音效的产生。使用射击标志位 m_shoot 来实现。见 playerLayer.cpp 内容实现。

游戏的大头，物理碰撞的思考：

下面 cocos2d 封装使用的 Physics 类的关键内容，因为它涉及到两物体之间的碰撞操作。通俗一点是分类操作，不同类有不同的碰撞反应及事件监听，当然这是人为操控的，下面就是控制分类的几个标志位介绍。

这里有三个值：**CategoryBitmask**，**ContactTestBitmask** 和 **CollisionBitmask**。您可以使用相应的 get/set 方法来获取/设置它们。它们是由逻辑和操作测试的。

当一个身体的 **CategoryBitmask** 并与另一主体的 **ContactTestBitmask** 其结果不等于零时，接触事件将被发出，相反接触的事件将不被发送。

当一个的身体的 **CategoryBitmask** 并与另一主体的 **CollisionBitmask** 其结果不等于零，则它们将碰撞，相反不会。

是独立的，在默认情况下，**CategoryBitmask** 值为 0xFFFFFFFF，**ContactTestBitmask** 值是 00000000，而 **CollisionBitmask** 值为 0xFFFFFFFF，表示所有的身体会相互碰撞，但默认不发送接触事件。

http://www.csdn123.com/html/topnews201408/50/450.htm 物理碰撞参考网址

再次注意的是 敌机只在可以视图才可以射击，在视图外它是不会发射子弹的。具体见敌人类的定时程式。

下面进入碰撞的行为定义：

	敌人	敌人子弹	玩家	玩家子弹
敌人	不会	不会	会	会
敌人子弹	不会	不会	会	不会
玩家	会	会	/	不会
玩家子弹	会	不会	不会	不会

下面定义各种碰撞行为

1: 玩家与敌人相撞　　玩家减血，直到 0 死亡

2: 玩家与敌人子弹　　敌人死亡

3: 玩家子弹与敌人　　玩家死亡

设计概念如下:

CategoryBitmask	CollisionBitmask	ContactTestBitmask
玩家　　: 0x00000001	玩家　　: 0x00000000	玩家　　: 0x0000000C
玩家子弹: 0x00000002	玩家子弹: 0x00000000	玩家子弹: 0x00000004
敌人　　: 0x00000004	敌人　　: 0x00000000	敌人　　: 0x00000001
敌人子弹: 0x00000008	敌人子弹: 0x00000000	敌人子弹: 0x00000001

碰撞的事件分配: 使用什么事件分配方式呢？

有如下几种

1: EventListenerPhysicsContact

2: EventListenerPhysicsContactWithBodies

3: EventListenerPhysicsContactWithGroup

4: EventListenerPhysicsContactWithShapes

解释: 1 将接收所有的碰撞回调

2 两个物体之间的特定碰撞

3 碰撞双方之一在组(Group)时的碰撞

4 两 shape 相撞时的碰撞

1. 碰撞开始调用;
   std::function<bool(PhysicsContact& contact)> onContactBegin
2. 两物体接触过程中运行， 返回 true 将正常运行碰撞处理， 返回 false 将视图出现 body 挤压变形现象，可以在这个函数里修改碰撞的相关值并使用 true 返回,
   std::function<bool(PhysicsContact& contact, PhysicsContactPreSolve& solve)> onContactPreSolve;
3. 两物体碰撞已经处理结束，可以处理 音效 及 伤害值等
   std::function<void(PhysicsContact& contact, const PhysicsContactPostSolve& solve)> onContactPostSolve
4. 在碰撞物体分离时将调用下行为：
   std::function<void(PhysicsContact& contact)> onContactSeperate;

现在困扰在如何组织碰撞注册及碰撞处理上面回调。具体代码见源代码。

下面介绍 Player 及 Enemy 销毁时伴随的行为

玩家销毁: 爆炸效果， 爆炸音效， 跳出结果层

敌人销毁: 将敌人精灵从层中删除，回收带武器的敌人， 销毁音效。这里解释一下敌人销毁的行为，试想敌人射出去的子弹 肯定不能随敌人的消失而消失。所以武器的不能绑定在敌人精灵上即 addChild 操作，庆幸的是我们的子弹是有半自动回收的功能，因为子弹是一定会移动到视图框外，那么由武器类就能够自动回收了。另外武器类是与图层相关的所有的，所以敌人控制是否发射子弹。消失以后即不会发射子弹了。但是在敌人层中的 scanSchedule(float Delta) 中的扫描是使用 EnemyWithWeapon 来确定的，所以移除已经使用的 m_UsedShips 不再拥有，下次不会有扫描，即不发再次发射子弹。

到这里大部分内容应该讲了，细节方面的还没有想到。

作者: 刘虎

时间: 2015/7/4