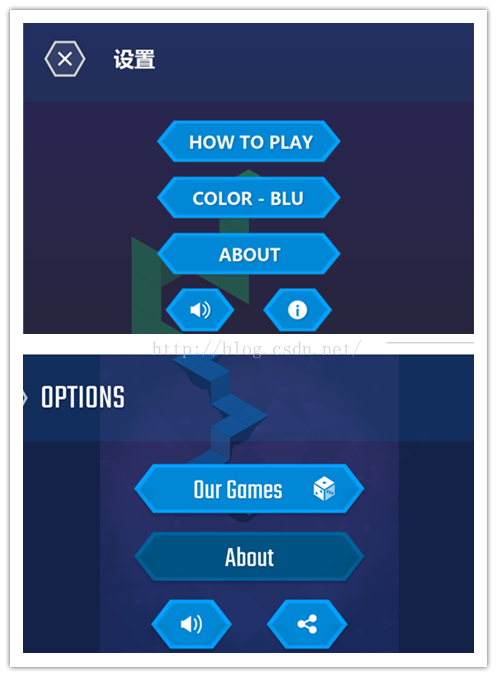
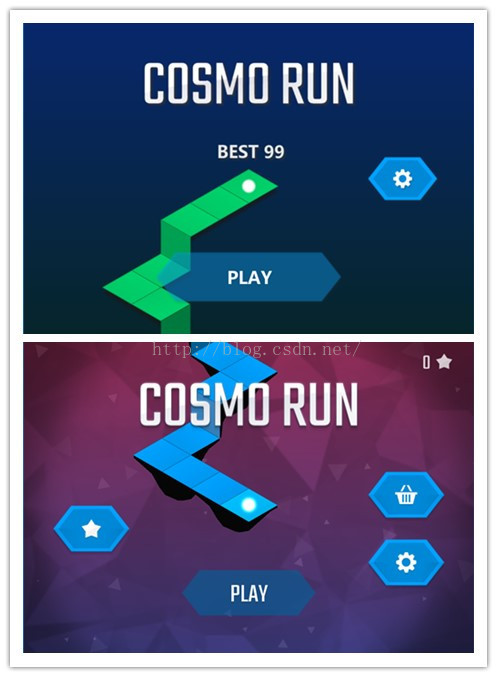
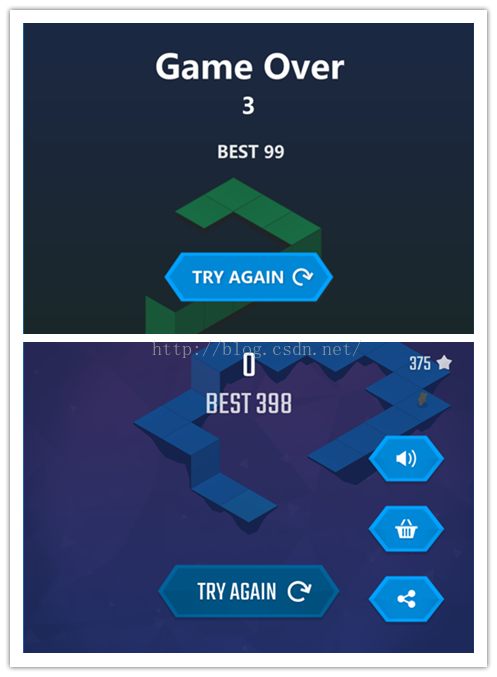
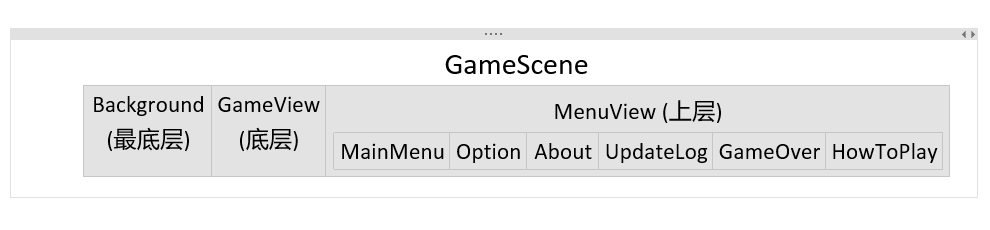
        首先，本人是java初学者，之前一直在学习C，看过一些C++的资料，用vc6.0 和 easyx写过几个小游戏。从5月初开始转战java，现在连面向对象的编程思想都没有学好，对游戏开发也无任何经验可谈。这个游戏完全仿照windows应用商店中的同名游戏制作，按照我自己对游戏开发的理解进行设计，目的是为了学习和熟悉javafx。

        下面是游戏与原作的对比图：





        游戏代码一共三千多行，框架很简单，基本结构如下表所示：

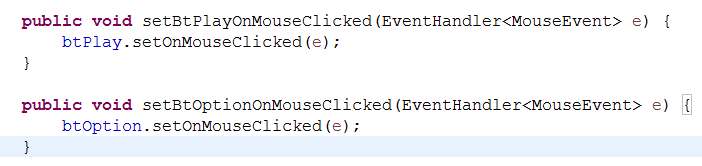


        每个view都封装在一个pane里，上层的六个界面用静态调用的方法互相切换，同时只会显示一个。

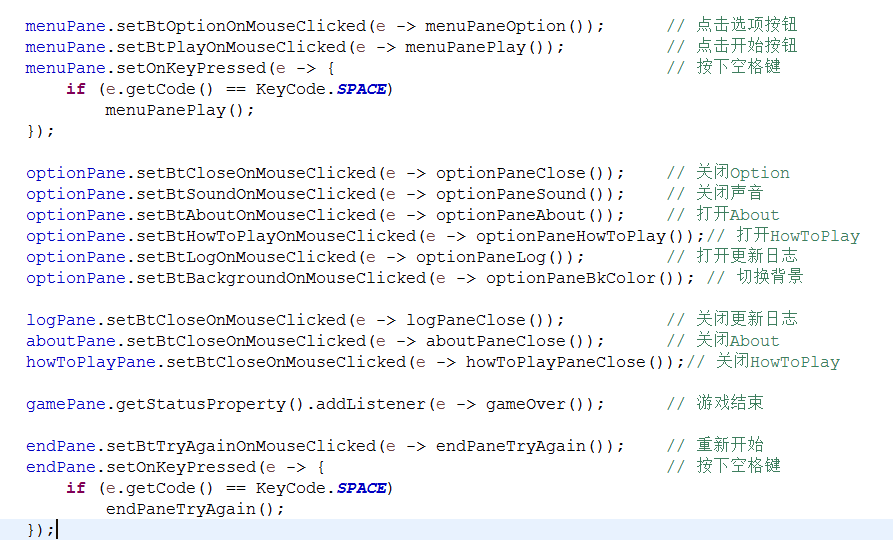
        这样做的好处是，从程序的任何一个地方都可以方便的访问到所有面板和它的数据，试想一下，如果不用静态方法，如何统一GameView、MainMenu、GameOver三个面板的得分和最高分？如何实现在MainMenu中点击OptionButton打开Option？

        第一个问题的解决方法是不断通过传递参数的方式访问储存在外部的得分，而第二个问题要先写一个public方法在外部控制OptionButton点击事件，再从外部用remove和add方法切换面板，也就是说，需要在外部搭一个桥。但是按钮有11个，这些按钮事件全都在外部处理时，要搭建至少11个桥，这样会出现极其冗长且难以阅读的代码，（然而我一开始确实是这样做的…）像下图这样

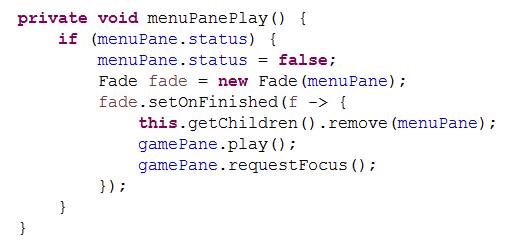
        这是岸...



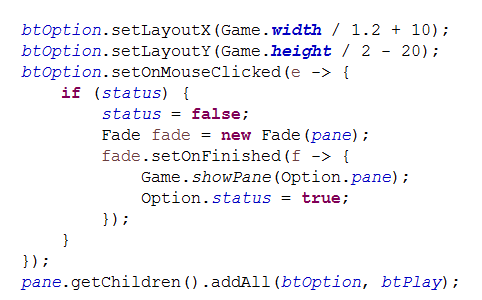
        这是桥...



          这是对岸...



        但是如果用静态调用的方法，只需要这几行就能实现



其中btOption、status、Menu.pane、Option.pane都是静态的。按钮点击事件和按钮的属性设置写在一起，可读性提高了很多。

PS.仔细一想，用静态和在C里定义全局变量有什么区别。。难道我是在用C的思想写java。。

上层View的结构统一是：半透明背景+ 文字 + 按钮，并封装起来。所有的按钮都有同一个父类HexagonButton（六边形按钮），按钮父类实现了设置长宽和鼠标移入、移出、按下、抬起时的变色。这里很重要的一点是，按钮是一个Group而不是一个Pane，很多初学者在封装时首先想到的是Pane，我学习的教材就从来没教我如何使用Group，而是一味的使用Pane，这是不对的。按钮上面的小图形也很简单，无非就是一些Circle、Rectangle、Polygon简单图形的组合。

这个游戏不管是图形，还是文字，都加入了BoxBlur虚化效果和DropShadow阴影效果，用effect.setInput(neweffect)方法同时加入这两种效果。  
   
接下来详细说明GameView部分，也就是游戏主体。  
  
 GameView 的界面有四部分，小球，板块，板块阴影和计分板。  
  
小球是一个简单的Circle类，固定在屏幕中间不动。  
小球有一个公用静态枚举 Direct，有六种可能值，标志小球左上、右上、左下、右下、直上、直下六种运动状态。  
小球可以做缩小和放大的ScaleTranslate动画。  
小球中有一个 Floor类，它是小球当前所在 Floor的引用。  
  
板块 Floor和板块阴影FloorShadow是大小相同颜色不同的 Polygon类，所有的 Floor 都在 Floors 这个 Group 中，所有的FloorShadow都在FloorsShadow这个 Group 中，之所以把Floor 和FloorShadow 分成两部分而不是把Shadow 当作Floor的一个属性处理，是因为板块是一块一块创建的，后创建的板块阴影会显示在前一个板块上面。  
Floor 有一个共用静态枚举MainMode，有三种可能值，标志Floor的三种形态。  
Floor 有一个共用静态枚举 Mode，有十二种种可能值，标志 Floor 的十二个方向。（Floor有 3 种形态，每种形态有4种方向）  
  
Floors 中有一个名叫hide的Floor类，顾名思义，它是一个隐藏的Floor，功能在后面介绍  
Floors 中的方法和用途如下：

int indexOf(Floor f)                    // 获得f的下标  
void add(Floor f)                        // 将f加入 Floors  
void remove(Floor f)                  // 将f从 Floors中移除  
int size()                                     // 获得当前 Floor的数量  
void clear()                                 // 清除所有 Floor  
void move(double x, double y)    // 通过 setTranslateX和setTranslateY整体移动所有Floor  
void hideAll()                              // 所有 Floor开始做缩小动画  
boolean isOutOfFloors()              // 小球出界返回 true   
boolean isCollisionWith(Floor f)  // f和其他Floor位置冲突返回true

GameView 有一个所有方法均为静态的 FloorManager类，它的主要方法和用途如下：  
Floor getFirstFloor()                   // 创建第一个 Floor  
Floor getNewFloor(Floor prev)   // 根据上一个Floor 类型，创建新 Floor  
void show(Floor f)                      // 让 f做放大动画  
void hide(Floor f)                       // 让 f做缩小动画  
  
GameView 同时做两种动画，moveAnimation和processAction  
moveAnimation 每 2毫秒，根据小球方向移动一次 Floors  
processAction 每 15毫秒，检查一次小球是否在当前的 Floor外，若不在当前Floor上，检查是否出界  
  
游戏开始时， FloorManager创建第一个板块，Floors将其设为“隐藏”的Floor（还记得那个名叫 hide 的Floor吗），然后运行15次addFloor()方法，这样就创建了16个板块。addFloor的过程是，用FloorManager创建一个新板块，检查这个板块是否和其他板块位置冲突，如果冲突，则重置“隐藏”的Floor，然后再运行一次addFloor，如果不冲突，则将隐藏的Floor加入Floors，然后重新新建一个隐藏的Floor。  
  
所以 hide 这个板块只是用来缓冲的，防止在板块形成一个圈时，无论怎么创建 Floor都会和其他Floor冲突，导致addFloor无限递归死掉。  
  
另外小球中还有 autoChangeDirect 方法，每次进入下一板块时调用，处理进入不同类型板块却不是转向的情况。  
  
 背景音乐用 java.applet.AudioClip 类的loop方法播放，这段音乐是我录制的游戏原版音乐。  
  
切换游戏背景时调用 Background.change() 和 Floors.setFloorColor()方法，都比较简单，我就不解释了。  
  
了解了游戏的运行模式，你可能会有以下疑问：  
Q1：为什么要用FloorManager创建新板块，而不是直接用Floor的构造方法创建呢？  
A1：创建新板块要考虑的东西很多，它的代码很长，将这段代码写在Floor的构造方法里可读性很差。而且FloorManager还有show和hide等方法，写了就知道，用这个类来管理Floor很爽。你会看到，Floor这个类极其简单和清晰。  
Q2：为什么不创建一个Floor数组，或者一个Floor的容器，非要定义一个Group的子类Floors？  
A2：Group的好处主要有三个，一个是它可以直接加到Pane里面，一个是它可以直接用setTranslate方法移动所有Floor，一个是它的contains方法可以直接判断一个点是否在它内部，判断小球是否出界的时候很好用。  
Q3：为什么播放音乐用AudioClip，不用javafx的media类？  
A3：因为media在循环播放音乐的时候经常莫名其妙停止，我没找到原因。  
  
最后贴上下载链接：  
  
游戏下载：链接：http://pan.baidu.com/s/1pLF21on 密码：7au8  
源代码下载：链接：http://pan.baidu.com/s/1nvw9Epb 密码：w8wf