综合实验报告

合作开发者

软件62 王兆伟 2016013254

软件62 谭新宇 2016010649

游戏介绍

我们的游戏名称为“接锅侠”，在每一局游戏中，画面上会有一个或多个篮筐与一系列道具，接下来会从手机界面的底部生成若干篮球，玩家通过在屏幕上画线，使得篮球进入篮筐，如果在球落回底部之前没有进入篮筐，则该局游戏失败，如果所有篮球都进入了篮筐，则玩家进入下一局。游戏画面的几个关键事物如下：

篮球 篮球在屏幕下方随机生成，但是它的位置与初速度随机。每一局中球的个数不超过3个。

篮筐 每局游戏存在随机数目与位置的篮筐，为了控制游戏难度，我们使得篮筐的数目不多于篮球的数目。我们尽力模仿篮筐为实际中的篮筐，它是一个兜形的物体，篮球无法从底部进入篮筐，而是会反弹。为了控制游戏合理性，球产生后不会径直撞向篮筐。

线 玩家在屏幕上触摸即可画线，每局游戏最多画三条线，可以把线看作一根刚性的链条，球撞到线上会产生反弹。

边框 手机屏幕除去下方的边框存在弹性较大的边框，当篮球碰到边框时会产生反弹效果。

星星 当篮球触碰到星星时，会获得加分，但是星星的存在时间是有时间限制的，且每颗星星的寿命随机（在2s到4s之间），随着星星的寿命减少，星星的颜色会逐渐暗淡，玩家可以通过这个变化来确定星星的寿命。

石头 石头是一个弹性较强的刚体，当球碰撞到石头时，会产生比较强烈的反弹效果。为了增加游戏难度，我们没有排除篮球会径直撞向石头的可能。

胜负判定 当有某个篮球没有进入篮筐而是落回屏幕底部时，游戏判负；当玩家画完三条线后一秒钟之后，存在篮球不在篮筐内，游戏判负。当玩家通过画线使得球永远不会落回底部也永远不会进框时，游戏将不会结束（事实上，玩家只需要用完画线的三次机会，游戏就结束了，我们不认为这是设计上的一个bug）。在其余情况下，游戏胜利，进入下一关。

界面布局设计

主域共有四个页面：游戏开始页面main，排行榜页面rankview，游戏进行页面game和游戏结束页面restart。由于提交了UI设计文件，故下面进行简要介绍。

- main：共有两个按钮和其他一些UI部件。**PLAY**按钮将开始游戏，即进入游戏进行页面，**排行榜**按钮将查看排行榜，即进入排行榜页面。其他一些UI部件是为了美观而添加。如背景，标题，篮球，指引游戏玩法的移动的手等等。
- game：固定组件有分数label与笔提示标志，其余为移动的刚体。由于刘海屏会挡住屏幕顶端的分数label，我们做了关于iPhone X的**刘海屏的适配。**
- restart：共有两个按钮和其他一些UI部件。**重新开始**按钮将重新开始游戏，即进入游戏进行页面，**回到主页**按钮将进入游戏开始页面。其他一些UI部件是为了美观和关系链数据而添加。如背景，在开放数据域绘制好的shareConvas来显示用户此次得分和历史最高分等等。
- rankview：共有一个按钮和其他一些UI部件，**返回**按钮将进入游戏开始页面。其他一些UI部件主要为了标识。如背景，标题，在开放数据域绘制好的shareConvas来显示排行榜等等。

  开放数据域只有一个页面GameRankingList。通过一些UI部件实现排行榜的优雅布局并将其绘制到shareConvas上从而在主域上渲染上屏。 此部分UI部件参考了该项目的布局：wxGameRank

游戏核心技术实现

我们的游戏开发平台是Cocos Creator。

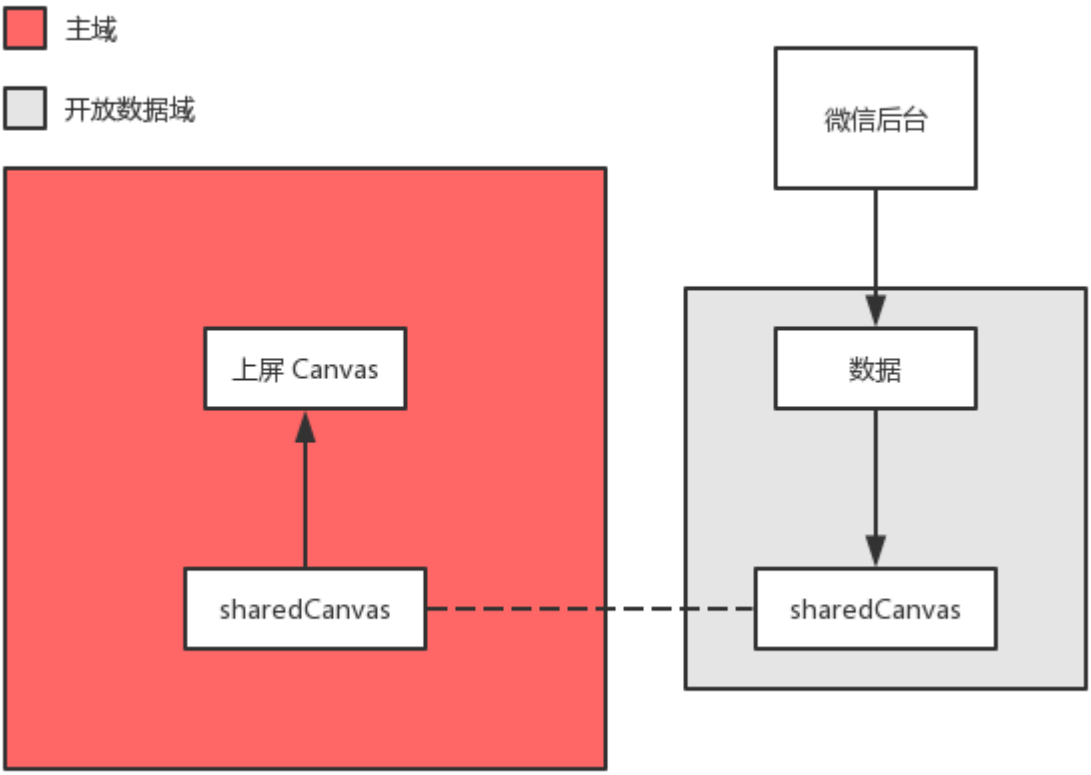
游戏的核心功能如下：玩家在屏幕上**画线**，且线实时更新为一根刚体，当球在碰撞到线上时，产生碰撞。

绘图 游戏的绘画部分不难实现，我们利用Cocos的引擎开发人员提供的绘图库**ccc.raphael**（其实就是利用了原生的moveTo等函数的写了一个更实用的库），即可实现在屏幕上绘图，具体实现如下：每一条线是一个Cocos中的node（节点），定义TouchStart，TouchMove与TouchEnd操作：TouchStart时，node中的points属性记录下来touch的坐标，当每次TouchMove时，将touch的坐标也传给points，也就是说，当每次TouchMove的时候，points列表都会得到更新，它记录的就是画线的轨迹。那么，怎么去实现画图呢？我们只需要在每次TouchMove的时候，调用库中的绘图函数，把这些点的连线绘制出来即可。

碰撞 Cocos引擎中有一个刚体的概念，含义就是实际的物理物体，当某些物体是刚体时，他们的碰撞效果由引擎提供，产生物理中碰撞的效果。我将篮球设置为一个球形刚体，问题是如何在绘制曲线的时候，使得曲线也是一个刚体？在最开始的时候，我们手撸了一段代码，用来根据绘图node中的points属性来计算得到曲线的包围多边形，后来我们发现，Cocos还提供了一种刚体称为Chain，即链条。我们只需要在每次TouchStart的时候，定义一根链条，然后每次TouchMove的时候，向链条中增加一个touch到的点，然后调用apply方法，即可更新链条。这样就实现了碰撞。

理想很丰满，现实很骨干。我发现了一个问题：代码写好以后，篮球不会和线产生碰撞。奇怪了！线已经是一个刚体了，为什么不会碰撞？我发现Cocos还有一个功能，可以显示刚体的实际轮廓，于是我打开这个功能，发现线根本不是刚体。奇怪了！我明明已经定义成了刚体啊！后来我才发现是坐标系的问题，touch的坐标与绘图的坐标是世界坐标系，但是由于刚体是一个定义在场景下的节点，它的坐标不应该是世界坐标，而应转化为她的parent节点下的局部坐标系，问题解决。核心技术实现。

排行榜实现



为了丰富游戏的社交玩法,微信小游戏开发文档中给出了获取微信用户关系链数据的API：

- wx.getFriendCloudStorage() 获取当前用户也玩该小游戏的好友的用户数据
- wx.setUserCloudStorage() 将当前用户的游戏数据托管在微信后台
- wx.getUserCloudStorage() 获取当前用户的托管数据

  我们使用了以上API来实现数据的获取和存取。 除此以外，要实现排行榜还需利用微信特有的主域和开放数据域的架构：

- 开放数据域不能向主域发送消息。
- 主域可以向开放数据域发送消息。调用 wx.getOpenDataContext()方法可以获取开放数据域实例，调用实例上的 OpenDataContext.postMessage()方法可以向开放数据域发送消息。
- 在开放数据域中通过 wx.onMessage() 方法可以监听从主域发来的消息。
- 只有开放数据域可以访问以上关系链的API。

  因此如果想要绘制排行榜，则需要创建两个项目，在开放数据域中将排行榜绘制到 sharedCanvas 上，再在主域将 sharedCanvas渲染上屏。 我们在主域上通过不同的情况向开放数据域发送消息，然后开放数据域通过不同的消息来做出相应的反应。如提交分数，如绘制排行榜，如绘制用户历史最高得分等等。 综上，通过以上两部分的结合，排行榜实现。

游戏重难点

除了核心技术，游戏中还有几件事等待我们解决：

1. 球最初产生时，初速度较快，如果球的初速度使得它径直撞向球网，就会产生较大的反弹速度，玩家根本来不及反应，游戏就输了。因此在初始化球的初速度的时候，我们不能使得它径直撞向球网。我设计了一个算法：球的初始化纵坐标by是固定的（在屏幕下方），横坐标进行随机，设值为bx，这时，每个球网被抽象为一个矩形，这样我们得到一个球网矩形的数组netrects。设球的y速度固定，我们来随机x速度，这时，我们根据球网矩形的位置和球的坐标（bx，by）可以得到一个球的初速度的范围，进而得到球的x速度的范围，我们对x进行5次随机，如果没有一个随机的结果使得x速度在合理范围内，那么我们认为这是由于球的bx没有随机好导致的，因此我们重新随机bx，重复以上操作。经过实验，一般最多只需要再随机一次bx就可以得到理想值，因此该算法是有效且省时的。从这里也可以看到，我们不是没有能力解决防止球径直撞到石头上，因为只需要把石头的包围矩形也加入netrects即可，而是为了增加游戏难度而没有这样做（滑稽）。

2.如何判断球进网了？我们在最开始提到，篮球是一个球形刚体，球网是一个兜形刚体（无盖），那么怎么判断球形进入兜形了呢？我在球网内部加入了一个传感器（Cocos已经实现了sensor），当球被传感器检测到时，证明球已经进了网中。

3.球进网会得分，撞到星星也会得分，但是在物理引擎中，有可能视觉上球进了一次网，但是在实际中有了几十次微小碰撞，这样吃一颗星星或者进一次网，就会得到很多分，那么，如何排除这一点呢？很简单，只需要立一个flag就好了。在球进网中，给球立一个flag，因为球最多只能进一个网，因此当球与网中传感器第一次碰撞后，就修改flag，使得下次碰撞不再积分即可。在球与星星相撞也是一样，因为一颗星星最多只能被吃一次。

游戏测试与数据设定

对iPhone X的刘海屏的适配问题 一方面刘海会遮挡住游戏界面的分数标签，另一方面，由于iPhone X的屏幕较窄，因此存在球网部分出现在屏幕外部的情况。为了便于助教测试，我们在游戏开始会判断手机是否是iphoneX或小米8，并对其进行特判。实际上若要正式发布应直接判断用户手机是否为刘海屏，而这需要相应的大量手机型号的信息，在此处我们不再进一步追究细节。

该不该随机关卡 在试玩同类游戏时，我们最初发现它的关卡设计十分合理，球的生成位置、速度以及球网的摆放和其它各类道具的摆放都十分合理。后来我们发现，是因为设计者已经提前设计了几张完美的图纸，然后每次开局都会从这几张图纸中选择一张。这样的后果是，用户在一开始可能觉得游戏十分好玩，但是只要多玩一两局，就会感到乏味，因为图纸的数目总是有限的，这不利于游戏的持久性。现在流行的微信小游戏，基本上都有一定的随机性来保证游戏的趣味性。因此，我们认为，可以通过略微减少图纸的合理性来增加游戏的趣味性。

球网的坐标设置与球的生成 在游戏重难点中已经介绍过球如何避开球网随机化算法。除此之外，我们尽量使得球的生成位置偏向屏幕中央，而球网的生成位置在屏幕两侧，这样的安排会使得玩家在画线时的手感更佳。通过测试，我们还发现了一个问题是，球网的生成位置有时会高于球的生成位置，尽管在这种情况下，玩家依然可以通过用线"顶球"的操作过关，但是我们认为这极大的降低了游戏的体验，因此我们修改了球网的生成参数，使其不能高于球的高度。

星星的坐标设置 星星作为加分项，一方面不能过于简单，比如星星离球网的位置过近，那么基本上进球就等于吃到星星，非常不合适。另一方面，如果星星与网的位置毫无关系，那么会导致玩家为了进球，一般不可能吃到星星。因此我们设计了星星会生成在篮网上方周围的区域。

石头的坐标设置 石头的设计纯粹是为了增大难度而诞生的。因为我们发现，在基于以上的设计中，过关似乎过于容易，因此我们将石头放在屏幕中央区域内，因为球的生成位置是在屏幕底部中央，因此球会有较大的概率撞到石头上，而且石头的弹性值较大，如果玩家的反应速度较慢，球就会迅速的反弹回底部，导致游戏失败。

一个没能解决的失误 我们发现，在测试中，如果用户画出的线缠绕在一起，就会出现较为严重的掉帧问题与性能问题（尽管少有用户进行如此操作），该现象在电脑上不会出现，但是在手机端受到性能的影响而出现，如果想要修改的话，可能需要修改游戏的核心算法，这里是我们在最初设计上的失误。

游戏亮点

- 画线进球 通过在图像上画线来让球进入，非常考验反应能力，在技术上的实现也很炫酷。
- 在游戏性的设计上，使得球最初生成时不是径直碰撞球网，实际上我们提出的是一个“线性规划”问题算法。
- 判定球进入球网时的碰撞检测，我们利用到了传感器，通过在网的内部放置传感器来检测球入筐。
- 将游戏中常出现的元素设为了prefab对象，从而对运行效率进行了优化。