设计文档（静态类图及相应的说明）

静态类图：



一、功能设计

（一）主要功能设计：

1）用户点击“开始游戏”按钮，进入画布弹球游戏界面；

2）通过移动“吞噬圈”，吞噬小球，累计得分（通过右上角显示）；

3）60秒后，游戏结束，弹出弹窗，得到用户最终得分；

4）用户可通过输入自己姓名，点击保存，保存游戏记录；

可查看游戏历史记录前三名的姓名、得分、时间信息；

可通过关闭弹窗，以游客方式继续游戏，不保存该次游戏记录。

（二）为方便测试添加和改变的功能：

1）显示累计得分时，同步显示剩余彩球数，方便观察彩球生成数量是否符合要求；

2）弹窗添加“清除历史记录”按钮，以方便清空记录，多次测试保存是否符合要求。

1. 代码设计：
2. 整体设计：

设计Shape类，Ball类、EvilCircle类继承Shape类，创建用户User类通过操作EvilCircle对象进行游戏。

1. 具体设计

1、游戏开始前，界面显示“开始游戏”按钮；按钮显示，画布隐藏；

2、用户点击“开始游戏”按钮，按钮隐藏（display=”none”），画布显示，调用startGame()函数开始游戏：

* startGame()函数设计：

1. 重置彩球数组balls = []; 重置彩球数count = 0; 重置累计得分score = 0;
2. 调用savingBalls()函数，生成并保存所有彩球；
3. 调用loop()函数，循环播放动画；
4. 通过setTimeout()函数，设置60秒时间后，弹窗显示（display=”block”），截取画布计分，显示得分，并调用outputFirstThreeUsers()函数输出排行榜前三名。

* savingBalls()函数：

1. 当balls数组长度小于50时，进入创建彩球循环；
2. 随机size=10或20 两种大小的彩球，随机v=2或7两种速度的彩球，随机位置、颜色，创建Ball彩球对象，将该对象加入balls数组中，记录彩球数量的count++；

注：为避免绘制错误，球至少离画布边缘球本身一倍宽度的距离；

为便于明显判断彩球速度快慢，设置彩球velX、velY都为随机选择的同一个速度v。

* loop()函数：

1. 设置黑色画板、画板尺寸；
2. 循环画出（draw函数）balls数组中exists==true的所有彩球，并用ball.update()更新彩球、ball.collisionDetect()检测彩球间的冲突；
3. 画出（draw函数）吞噬圈，并用evil.checkBounds()进行边缘检测、evil.collisionDetect()检测彩球与吞噬圈之间的冲突，通过调用scoring()方法累计得分。
4. 通过 requestAnimationFrame(loop)函数设置动画循环播放。

* outputFirstThreeUsers()函数

1. 取存储到本地的数组到localtionUsers数组；
2. 若本地存储的数组为null时，为localtionUsers新建一个空数组，以作差错处理（测试修改得）；
3. 通过locationUsers.sort(function (a, b) {return b.ascore - a.ascore;})对locationUsers数组按成绩高低进行排序；
4. 在表格中输出小于locationUsers数组长度的前三名，通过创建”tr”标签元素，填入数据，加入表格以作显示，具体代码如下：

//在表格中输出前三名

for (let i = 0; i < locationUsers.length && i < 3; i++) {

var tr = document.createElement("tr"); //创建HTML"tr"标签元素

tr.innerHTML = "<td>" + locationUsers[i].name + "</td>"

+ "<td>" + locationUsers[i].ascore + "</td>"

+ "<td>" + locationUsers[i].date + "</td>"; //将数组内容填入新创建的tr元素

var ttable = document.getElementById("history");

ttable.appendChild(tr); //将创建的tr元素加入显示表格

}

* Scoring()函数实现计分，当彩球快小计20分，慢小计15分，快大计10分，慢大计5分。
* 注：loop()函数中update()、collisionDetect()函数，evil.checkBounds()、evil.collisionDetect()函数都为老师示例代码所给做简单修改，只在以下说明大概实现过程与功能。
* update()为定义彩球更新函数，当彩球碰到画布边界，方向调转，速度大小不变；
* collisionDetect()为定义碰撞检测函数，当存在两彩球圆心之间的距离小于其半径之和，即为碰撞，彩球碰撞后，两彩球随机一个相同的颜色；
* evil.checkBounds()为定义 EvilCircle 的边缘检测方法，防止吞噬圈移出屏幕；
* evil.collisionDetect()// 定义 EvilCircle 冲突检测函数，存在的彩球与吞噬圈圆心之间的距离小于其半径之和，该彩球设为不存在，count--,得分累加；

1. 游戏结束，在以上startGame()函数内，弹框显示得分、排行榜等，用户可通过向弹框中的文本框输入姓名，点击“保存”按钮，作响应：调用savingUser()函数保存用户信息到本地，后使用location.reload()返回游戏开始界面。

* savingUser()函数

1. 通过JSON.parse(localStorage.getItem("messageStorage"))取本地存储的数组到localtionUsers；
2. 当本地存储的数组为null时，为localtionUsers新建一个数组；（测试修改得）
3. 取当前用户输入的姓名、成绩、及获得成绩的时间，建立User对象加入到locationUsers数组；
4. 通过 localStorage.setItem("messageStorage", JSON.stringify(locationUser)将locationUsers数组存入本地;

1. 为方便测试检测，用户可在弹窗界面点击x关闭弹窗（隐藏），以检测弹球数量与大小速度等得分标准无误；可点击“清除历史记录”调用locationStorage.clear()清除本地存储，然后返回游戏开始界面，以便于重复多次测试，数组排序显示问题。

总结：

1）函数设计尽量封装，但不知多个函数未封装到类里，OPP风格设计存在改进可能；

2）要求功能基本实现，添加部分功能以便于测试检测；

3）某些易用性、观赏性、实用性的设计存在改进可能；

3）设计实现中，复习了JavaScript语法知识，熟悉了JavaScript的应用，代码编写能力、程序设计能力有了较大部分的提升，成就感很大。