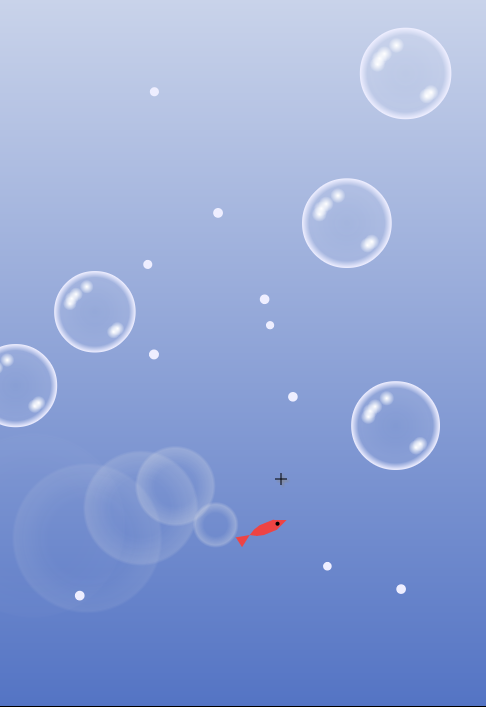
前端课程综合实验说明文档

创意游戏设计



周伯威|2012012221|zhou\_bw@yeah.net

林杨湄|2012013279|linym012@163.com

# 目录

1. 游戏介绍

1.1 游戏方法

1.2 子弹介绍

1.3 小鱼介绍

1.4 技能介绍

2. 设计思路

2.1 游戏类型设计

2.2 子弹种类设计

2.3 游戏的名字

3. 实现过程

3.1 时钟周期

3.2函数编写

3.3 特殊技能的实现

3.4 子弹移动

3.5 碰撞事件判定

3.6 难度均衡化

3.7 背景音乐选择

4. 技术难点

4.1 完全使用canvas绘图

4.2 小鱼的运动特性

4.3 背景颜色的连续变化

5. 其他亮点与技术细节

# 1. 游戏介绍

游戏发布链接为<http://zbww.github.io/fe/g/>

## 1.1 游戏方法

移动鼠标控制屏幕上的小鱼，需要躲避飞来的各种物体(下称"子弹")。若吃到钻石，可以随机获得一种特殊能力。游戏并没有尽头，设有8个level，难度依次递增。开始时小鱼共有五条生命，当这些生命耗尽时游戏结束。

## 1.2 子弹介绍

玩家在游戏中共会遇到四种子弹，下面逐一介绍其特性(括号内为其在代码中的代号):

① 小泡泡(ball): 半径为4~5.2像素，颜色为#eef的圆形，在屏幕右侧边界随机生成，运动方向为左方±5°，速度适中，密度大，较容易躲避，每个level都会出现。

② 大泡泡(bigBall): 半径为40~48像素，颜色为#eef的圆形，运动特性与小泡泡相同，但速度略慢，密度低，躲避难度中等，在2、5、6、MAX等级出现。

③ 蝴蝶(butterfly): 看起来很大的一种子弹，但实际判定点为中心的一个圆形，半径为8像素，蝴蝶的颜色使用一定规则随机生成，路径为圆弧，速度快，密度低，躲避难度高，在3、6、7、MAX等级出现。

④ 星星(star): 黄色的六角星，判定点是半径为6的圆形，路径为直线，但方向为追踪小鱼的方向，速度快，密度中等，躲避难度高，在4、5、7、MAX等级出现。

## 1.3 小鱼介绍

小鱼(plane)是一个红色的多边形，眼睛为黑色圆形，尾部可摆动，身后会出现水波纹，以上特点在下文中将详细说明。除此之外，小鱼的中弹判定点是和小泡泡大小相近的圆形。

## 1.4 技能介绍

游戏过程中共可随机获得六种技能，现列举如下:

① Score++ (func\_addScore): 分数随机增加3000~6000。

② Speed Down (func\_slow): 将当前屏幕的子弹和即将出现的子弹速度减半，一个重要细节是子弹的生成速度也将减半，否则全屏幕将充满先前的子弹。

③ 1 UP (func\_oneUp): 生命值加一。

④ Superfish (func\_wudi): 无敌6000毫秒，此期间小鱼出现保护罩，同时闪烁。

⑤ Big Bomb (func\_clear): 清除当前屏幕的子弹。

⑥ Mini World (func\_small): 除了小泡泡外的三种子弹以及小鱼，外观与判定点的尺寸均减半。

# 2. 设计思路

## 2.1 游戏类型设计

经过一下午的思考，我们决定制作一款弹幕射击类游戏。在最初的版本中，玩家控制的对象被设计为飞机形状，机头可以朝向各个方位，子弹也被设计成随机位置与随机方向，此外，飞机本身也可进行射击。但这样一来，游戏的结构混乱，可玩性很低，我们便将其类型定位为横版弹幕躲避类游戏，同时也将飞机换成了小鱼的形状。

## 2.2 子弹种类设计

既然是弹幕躲避类游戏，就必然要设计多种不同的子弹。为了使攻击没有死角，除星星外的子弹弹道、速度均为随机生成。在只有两种泡泡的时候，玩家只需上下躲避即可，为避免这种玩法，我们设计了蝴蝶弹，其运行轨迹为上下交错的圆弧，蝴蝶弹的出现使游戏难度大大增加。星星弹是追踪小鱼发射的，这样设计使得小鱼不能在一定范围内过久停留。

虽然只设计了四种子弹，但其丰富的特性足以使游戏的可玩性达到专业水准。

## 2.3 游戏的名字

不是我想的。

# 3. 实现过程

## 3.1 时钟周期

在本游戏中，时钟(clock)是很重要的一个全局变量。

游戏采用了60fps的绘图速率，每一帧使clock加一。很多函数，比如背景变化、鱼尾摆动、子弹生成均与该时钟有关。下面举例说明:

① 鱼尾摆动: 是随时钟做正弦运动的，其角度为8\*cos(7\*clock)。

② 子弹生成: 两种泡泡为随机生成，在每个clock周期内将生成一个随机数，若该随机数小于某值则添加子弹。蝴蝶与星星均为每隔一定时间生成一次，星星出现的位置还随clock做正弦变化。

## 3.2函数编写

此次实验的代码中，函数均采取了一定规则来书写:

① 画单个子弹的函数: 形如drawOneBall()

② 画同一类子弹的函数: 形如drawBalls()

③ 添加新子弹的函数: 形如addBall()

④ 计算子弹移动的函数: 形如ballMove()

⑤执行某种特殊技能的函数: 形如func\_small()

这样，游戏对于不同类型的子弹、技能能够采取统一的方式书写函数，也方便了新子弹/技能的添加。

在代码的前半部分，有一个预处理函数，该函数能保证在每次重新开始时能够完全重置变量等。

## 3.3特殊技能的实现

① 加分: 直接增加全局变量score的数值。score表示额外增加的分数，最终的分数为10\*(score+clock)。

② 速度减慢: 代码对于子弹的速度控制，有形如ballSpeed的全局变量，速度减慢会使该变量减半。对于已经生成的子弹，速度减慢调整的是已有子弹的speed参数(对于蝴蝶是角速度参数)。此外，形如ballDensity的控制子弹生成密度的全局变量也应减半。使用计时器来控制减慢效果持续时间。

③ 生命增加: 直接增加全局变量life的数值。

④ 无敌: 增加计时器，取消鱼的中弹判定，时间满后恢复。与无敌同时出现的光环、闪烁现象则靠全局变量wudi来控制绘图函数做出相应调整。

⑤ 清屏: 清空当前子弹数组即可。

⑥ 缩小: 修改控制子弹、鱼的大小的全局变量，并重新生成各个图形的形状参数。使用计时器控制效果持续时间。

## 3.4 子弹移动

为了产生动画效果，每一帧重绘时，子弹都应移动位置，这时便需要逐一进行位置的计算。对于两种泡泡以及星星，子弹的新位置即是其旧位置加上相应的速度。蝴蝶弹的运动轨迹是圆弧，其圆心固定，则新的方位角为旧的方位角加上角速度，以此来计算新位置。

重新计算完所有子弹的位置后便可交由绘图函数进行绘制。

## 3.5 碰撞事件判定

上文提到，本游戏中，所有物体的判定点均为圆形。采取这种方法，既降低了躲避难度又降低了代码编写难度。

为节约时间提升效率，碰撞事件并不是在每一帧内单独循环一次计算，而是加在了物体移动函数(如ballMove)中进行: 枚举当前屏幕中，某类型的全部子弹，判断中心点到鱼中心点的距离是否小于临界距离。如果距离足够小，则执行kill()函数，进行进一步的判断。

## 3.6 难度均衡化

游戏大致制作好的时候，小鱼的生命只有一条，而且关卡的设计是每一关都新增加一种子弹，第四关即是四种子弹同时出现。显然，这样的设计过难，无法带来较好的游戏体验。

为了降低难度，我先是将四种子弹分散到了八个等级中，在最后一级才使它们同时出现。然后降低了各种子弹的参数，其中包括小泡泡的密度、蝴蝶的数量、蝴蝶的角速度、星星的大小、速度以及密度。与此同时，我也在一些地方尝试着增加了难度，包括加快泡泡速度、使星星瞄准小鱼的角度有0/+5/-5度的偏差。

至最终版本时，游戏难度已十分均衡: 刚上手的玩家可以玩到level 3左右，比较熟练的玩家则可以玩到level MAX。

## 3.7 背景音乐选择

背景音乐非原创，为游戏《Child of Light》中的音乐《Pilgrims on a Long Journey》。

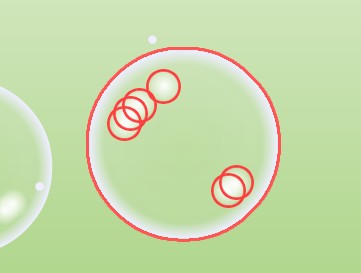
确定关卡时间时也考虑了背景音乐的因素: 当进入第二关时恰好为背景音乐节奏加快的时候。

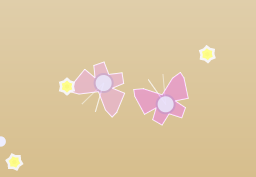
此外，鱼身后的水波出现频率与音乐频率一致，在不同时间有不同的调整。

# 4. 技术难点

## 4.1 完全使用Canvas画图

本游戏中，未使用任何外部图像资源，游戏界面的所有图形均为Canvas绘制，下面选择有代表性的进行说明:

① 小鱼: 以鱼身中心点为极点，鱼头方向为极轴，使用极坐标描述鱼的轮廓上各点的位置信息。使用极坐标的好处是便于旋转图像。鱼尾根据clock做周期性摆动，摆动角度为正弦函数。绘制摆动的鱼尾的方法同样为极坐标，将两点分别加减一个角度；为消除此方法造成的鱼尾的变形，我让鱼尾两点到极点的距离也做了正弦的变化。

② 大泡泡: 右面给出了画法的示意图。泡泡上的光斑较难绘制，我使用了如图所示的方法，将光斑拆解成六个圆形，每个圆形分别填充上径向渐变即可达到效果。

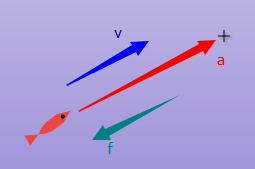
③ 蝴蝶: 绘制方法类似于小鱼，使用了极坐标确定位置。蝴蝶的颜色采取如下随机方法: 取100~244的随机数t，将t与244与255三个数随机分配给R、G、B。透明度alpha取0.4。这种随机方法使得蝴蝶颜色十分柔和。

④ 钻石: 由于钻石不涉及到旋转，其绘制并未采用极坐标方法而使用了直角坐标。为了使钻石看起来更像可以吃的道具，我给它加上了周期性的闪烁，即，钻石的透明度随clock做正弦变化。

## 4.2 小鱼的运动特性

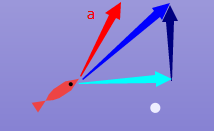
体验过本游戏后可以发现，使用鼠标操作小鱼可以获得非常好的体验。那么，这是如何实现的呢?

最初的设想是使小鱼和鼠标位置完全相同，但这样的游戏效果很差，运动真实性低。我便尝试将鱼的加速度设置为鱼头部(注意鱼的位置判定为鱼身中间)到鼠标指针的距离值，就像鼠标与鱼之间有一根弹簧一样。但这样一来鱼会绕着鼠标做简谐振动，无法稳定下来。

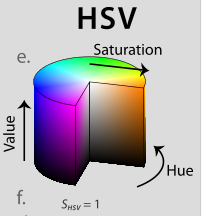


联系到物理学到的知识，我尝试对该振动添加一个过阻尼，即，添加一个与速度成正比且反向的阻力f，通过调整比例，最终达到了完美的效果，鱼可以加速游动到指针位置，又不会游过。

早先版本的鱼头是可以360°旋转的，但这不符合游戏中鱼向右游动的设定。因此，我使用了这样的模型: 取加速度的竖直分量(图中#000080)，将该向量与一水平定长向量(#00FFFF)相加，得到的向量(#0000FF)即当做鱼头方向。这种方法能使鱼头十分平稳。



## 4.3 背景颜色的连续变化

早期版本中，背景颜色选取为比较单调的浅蓝色到深蓝色的渐变。我们认为使用颜色变化的背景比较符合"梦"的主题。但这样就面临一个问题: 构建时间到颜色的一个函数，使得颜色能随时间均匀变化。

我们想到了HSV颜色模型，固定饱和度与明度，使色相从1°到360°变化，这样就实现了颜色连续变化。我们的背景采取线性渐变，两个端点所固定的(S, V)分别为(0.14, 0.92)与(0.57, 0.77)，该渐变色看起来较为柔和。

# 5. 其他亮点与技术细节

* 游戏界面充满整个浏览器，且子弹密度并不会随分辨率改变而改变。
* 小鱼身后会出现水波，水波随时间变大、变透明，且运动速度与小泡泡速度相当。水波频率与音乐频率有关，这是由时钟控制实现的。
* 为增大游戏难度，钻石的掉落位置并不是均匀分布的，而是靠右侧分布较密集。
* 小鱼靠近左上角时，左上角的分数栏会逐渐变透明。
* 游戏可以暂停，暂停时，clock变量停止。
* 使用localStorage来存储分数与排名。
* 对于大泡泡与蝴蝶，由于采用了复杂的绘图函数，在每一帧(16毫秒)不能大量绘制，故采用了较少的这些子弹。
* 控制无敌的计时器开始前要清除已有计时器，否则上一个计时器停止时会强制结束无敌状态。但这一问题对变小、变慢无影响。
* 使用清屏、变小、变慢这三个道具时，我采用将屏幕闪烁两下的方式来提醒玩家游戏的变化。实现方法是将全局变量"flash"设置为8，在绘图的时钟里，如果flash为1、2、7、8，则将屏幕绘制为白色，如果flash>0则将其减一。这样便在8个时钟周期完成两次闪烁。
* 本游戏除背景音乐外，均为原创。

*祝玩得愉快!*