HTML5 实验室

——Canvas 世界

张　磊　编著

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书通过多个实验，由点到面地对 HTML5 相关的技术进行详细的介绍和剖析，涵盖了 HTML5 标准中描述的新特性。本书以简洁的文字，结合生动的实验介绍 HTML5 的特性，并深入剖析其内部原理，讲授如何使用 Canvas 集成特定的算法去实现绚丽的效果、应用和游戏，其中涵盖了数学和物理方面的知识，让读者不仅知其然，而且知其所以然；最后通过几个综合实验和经典游戏的重现，将各种新特性综合，实现酷炫的网页效果。

阅读本书不需要预先具备特定编程语言的知识，任何渴望投入 HTML5 世界的新手，想要巩固和复习数学和物理知识的编程老手，以及具备其他语言的编程经验（如 JavaScript、C#），想要了解 HTML5 新特性或加深对编程语言理解的程序员都适合阅读本书。

图书在版编目（CIP）数据

HTML5 实验室：Canvas 世界/张磊编著. —北京：电子工业出版社，2012.6

ISBN 978-7-121-17157-4

Ⅰ. ①H… Ⅱ. ①张… Ⅲ. ①超文本标记语言 ②网页制作工具 Ⅳ. ①TP312 ②TP393.092

中国版本图书馆 CIP 数据核字（2012）第 106624 号

责任编辑：窦 昊
印　　刷：北京丰源印刷厂
装　　订：三河市鹏成印业有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：16　字数：358 千字
印　　次：2012 年 6 月第 1 次印刷
印　　数：4000 册　　定价：49.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

2005 年，许多网站相继加入 AJAX 应用。AJAX 是可以创建更好、更快和交互性更强的 Web 应用程序的技术，基于此技术，互联网应用程序可以变得更完善、更友好。AJAX 最开始代表异步请求和局部刷新，后来演变成为网页里一切好玩的东西，它拉近了 Web 程序与桌面程序的距离。

到了 2007 年，W3C 针对是否接纳 HTML5 进入标准审核程序的提议进行了投票，多数人表示赞成。后来，W3C 承认 HTML5 是正式标准。HTML5 是近 10 年来 Web 标准最巨大的飞跃，和以前的版本不同，HTML5 并非仅仅用来表示 Web 内容，它还提供了 Canvas、WebGL、Audio、Notifications、LBS、WebSocket、SVG、Storage、Web Workers、IndexDB、FileAPI 等重要元素，每个标签都有其相应的开源项目，且可以独立成书。HTML5 的使命是将 Web 带入一个成熟的应用平台，在这个平台上，视频、音频、图像、动画、存储，以及与计算机的交互等都被标准化。如果说 AJAX 是在拉近 Web 程序与桌面程序的距离，那么 HTML5 即将抹去 Web 程序与桌面程序的差距，没人能阻止开发者将桌面程序搬到网页里。Web APP 的时代已经来临，HTML5 正在重塑一个崭新、绚烂的 Web 世界。

本书使用 HTML5 的 Canvas 作为实验平台，JavaScript 为编程语言，进行了大量的粒子实验、物理实验、3D 实验、像素实验和文字实验，然后带着读者从实验走向实战，带领读者一步步制作一个物理引擎，最后带领读者制作一款完整的 HTML5 游戏。

本书介绍的每个实验都可以移植为 Java、Objective-C、C++或者 ActionScript 版本，也可以移植到 XNA 或者 Silverlight 上，等等。所以本书面向的读者，不仅仅是前端开发工程师，也适合游戏开发人员、大学生或者高级美工阅读。本书同样适合对视觉艺术、计算机图形学、物理、数学（特别是线性代数）感兴趣的读者。本书所有代码都经过严格的测试和反复的使用，读者可以放心使用，如有任何问题及疑问，请邮件联系 mHTML5@qq.com。

关于本书代码及演示

本书是一本以实践为目标的图书，包含大量的物理、数学和计算机图形学的集成实验，每个实验的设计思想以及核心算法均辅以对应的代码示例。本书不包含任何伪代码算法描述，全部代码均采用 JavaScript 语言加以实现。

本书代码提供了理解算法问题所必需的细节，并辅以大量的分析图片，展示了解决问题的核心推导步骤，读者可以在支持 HTML5 的浏览器（如 IE9 及以上版本、火狐浏览器、谷歌浏览器、Opera 等）中运行相关代码并查看呈现结果。读者也可以改变代码中的一些核心参数或者变量的值，然后执行代码，查看其结果的变化，这对于不熟悉算法、数学思想和调试的读者尤为重要。

虽然本书的代码示例采用的是 JavaScript 语言，但是需要强调的是，其他计算机语言都可以完成本书介绍的所有实验。需要注意的是，本书中的示例代码并非最终版本，虽然所有代码都经过反复使用和严格测试，但是不能保证其没有重构的空间，读者在理解其核心思想和架构的基础上可以自己进行相应的重构。

本书中的所有代码按照章节依次分类，可在电子工业出版社官网（www.phei.com.cn）下载，也可以向编辑索要（yangbo2@phei.com.cn）。书中介绍的每个实验都包含一个或者多个演示文件，详细展示了整个实验的制作过程，让读者循序渐进地理解其算法和思想。读者可以用各种文本编辑器

或者 IDE（如 notepad、notepad++、Visual Studio、sublime text、aptana studio、Web Developer Express 或 Expression Web 等）打开查看。

作　者
2012 年 4 月

目　录

CONTENTS

上篇　实　　验

下篇　游 戏 开 发

上篇

实　验

第 1 章　基 础 实 验

实验 1　奥 运 五 环

画圆

```
arc(x, y, radius, startAngle, endAngle, counterclockwise)
```

```
<canvas id="myCanvas" width="400" height="200" style="border: 1px
solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>
    <script type="text/javascript">
        var c = document.getElementById("myCanvas");
        var ctx = c.getContext("2d");
        ctx.beginPath();
        ctx.arc(70, 70, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.beginPath();
        ctx.arc(160, 70, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.beginPath();
        ctx.arc(250, 70, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.beginPath();
        ctx.arc(110, 110, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.beginPath();
        ctx.arc(200, 110, 40, 0, Math.PI * 2, false);
        ctx.stroke();
    </script>
```

```
        var c = document.getElementById("myCanvas");
        var ctx = c.getContext("2d");
```

```
        ctx.lineWidth = 5;
        ctx.strokeStyle = "#163B62";
        ctx.beginPath();
        ctx.arc(70, 70, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.strokeStyle = "#000000";
        ctx.beginPath();
        ctx.arc(160, 70, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.strokeStyle = "#BF0628";
        ctx.beginPath();
        ctx.arc(250, 70, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.strokeStyle = "#EBC41F";
        ctx.beginPath();
        ctx.arc(110, 110, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.strokeStyle = "#198E4A";
        ctx.beginPath();
        ctx.arc(200, 110, 40, 0, Math.PI * 2, false);
        ctx.stroke();
```

画弧

```
        arc(x, y, radius, startAngle, endAngle, counterclockwise)
```

```
        ctx.strokeStyle = "#163B62";
        ctx.beginPath();
        ctx.arc(70, 70, 40, Math.PI * 1.9, Math.PI * 2.1, false);
        ctx.stroke();
        ctx.strokeStyle = "#000000";
        ctx.beginPath();
        ctx.arc(160, 70, 40, Math.PI * 0.9, Math.PI * 2.1, false);
        ctx.stroke();
        ctx.strokeStyle = "#BF0628";
        ctx.beginPath();
        ctx.arc(250, 70, 40, Math.PI * 0.9, Math.PI * 1.1, false);
        ctx.stroke();
```

```
  <canvas id="myCanvas" width="400" height="200" style="border: 1px solid #c3c3c3;">
```

```
Your browser does not support the canvas element.
</canvas>
    <script type="text/javascript">
        var c = document.getElementById("myCanvas");
        var ctx = c.getContext("2d");
        ctx.lineWidth = 5;

        ctx.strokeStyle = "#163B62";
        ctx.beginPath();
        ctx.arc(70, 70, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.strokeStyle = "#000000";
        ctx.beginPath();
        ctx.arc(160, 70, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.strokeStyle = "#BF0628";
        ctx.beginPath();
        ctx.arc(250, 70, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.strokeStyle = "#EBC41F";
        ctx.beginPath();
        ctx.arc(110, 110, 40, 0, Math.PI * 2, false);
        ctx.stroke();
        ctx.strokeStyle = "#198E4A";
        ctx.beginPath();
        ctx.arc(200, 110, 40, 0, Math.PI * 2, false);
        ctx.stroke();

        ctx.strokeStyle = "#163B62";
        ctx.beginPath();
        ctx.arc(70, 70, 40, Math.PI * 1.9, Math.PI * 2.1, false);
        ctx.stroke();
        ctx.strokeStyle = "#000000";
        ctx.beginPath();
        ctx.arc(160, 70, 40, Math.PI * 0.9, Math.PI * 2.1, false);
        ctx.stroke();
        ctx.strokeStyle = "#BF0628";
        ctx.beginPath();
        ctx.arc(250, 70, 40, Math.PI * 0.9, Math.PI * 1.1, false);
        ctx.stroke();
    </script>
```

设置圆环的宽度

设置圆环的颜色

从这里开始，下面是画弧

实验 2　台　球

实心圆

```
<canvas id="myCanvas" width="200" height="100" style="border:1px
solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>

<script type="text/javascript">
    var c = document.getElementById("myCanvas");
    var cxt = c.getContext("2d");
    cxt.fillStyle = "#FF7F00";
    cxt.beginPath();
    cxt.arc(70, 18, 15, 0, Math.PI * 2, true);
    cxt.closePath();
    cxt.fill();
</script>
```

```
    var parentBall = {
        position: { x: 200, y: 200 },
        r: r,
        color: "#fff"
    };
    var firstBall = {
        position: { x: 550, y: 200 },
        r: r,
        color: "#fff"
    };
```

```
    for (var j = 1; j < 5; j++) {
        for (var k = 0; k < j + 1; k++) {
            var ball = {
                position: { x: firstBall.position.x + j * Math.sqrt(3) * r, y:
firstBall.position.y - j * r + 2 * k * r },
                r: r,
```

```
                color: "#fff"
            };
            balls.push(ball);
        }
    }
```

```
    cxt.fillStyle = "#fff";
    for (i in balls) {
        cxt.beginPath();
        cxt.arc(balls[i].position.x, balls[i].position.y, balls[i].r, 0, Math.PI *
2, true);
        cxt.closePath();
        cxt.fill();
    }
```

```
    <canvas id="myCanvas" width="847" height="440"
style="background-color:#000000">
Your browser does not support the canvas element.
</canvas>
<script>
    var canvas = document.getElementById("myCanvas");
    var cxt = canvas.getContext("2d");
    var r = 10;
    var balls = [];
    function initBall() {
        var parentBall = {
            position: { x: 200, y: 200 },
            r: r,
            color: "#fff"
        };
        var firstBall = {
            position: { x: 550, y: 200 },
            r: r,
            color: "#fff"
        };
        for (var j = 1; j < 5; j++) {
            for (var k = 0; k < j + 1; k++) {
                var ball = {
                    position: { x: firstBall.position.x + j * Math.sqrt(3) * r,
y: firstBall.position.y - j * r + 2 * k * r },
                    r: r,
                    color: "#fff"
```

initBall 方法生成所有的小球。

根据 firstBall 的坐标，通过两个嵌套的 for 循环生成所有小球坐标

```
                };
                balls.push(ball);
            }
        }
        balls.push(firstBall);
        balls.push(parentBall);
    }
    function draw() {
        cxt.fillStyle = "#fff";
        for (i in balls) {
            cxt.beginPath();
            cxt.arc(balls[i].position.x, balls[i].position.y, balls[i].r, 0,
Math.PI * 2, true);
            cxt.closePath();
            cxt.fill();
        }
    }
    initBall();
    draw();
    </script>
```

```
    function randomColor() {
        var arrHex = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B",
"C", "D", "E", "F"]; var strHex = "#";
        var index;
        for (var i = 0; i < 6; i++) {
            index = Math.round(Math.random() * 15);
            strHex += arrHex[index];
        }
        return strHex;
    }
```

```
    var firstBall = {
        position: { x: 550, y: 200 },
        r: r,
        color: randomColor()
    };
    for (var j = 1; j < 5; j++) {
        for (var k = 0; k < j + 1; k++) {
            var ball = {
                position: { x: firstBall.position.x + j * Math.sqrt(3) * r, y:
```

产生随机颜色

```
firstBall.position.y - j * r + 2 * k * r },
                r: r,
                color: randomColor()
            };
            balls.push(ball);
        }
    }
```

产生随机颜色

```
    function draw() {
        for (i in balls) {
            cxt.fillStyle = balls[i].color;
            cxt.beginPath();
            cxt.arc(balls[i].position.x, balls[i].position.y, balls[i].r, 0,
Math.PI * 2, true);
            cxt.closePath();
            cxt.fill();
        }
    }
```

在这里使用

实验 3　绘 制 动 画

画线

```
<canvas id="myCanvas" width="200" height="100" style="border:1px
solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>

<script type="text/javascript">
    var c = document.getElementById("myCanvas");
    var cxt = c.getContext("2d");
    cxt.moveTo(20, 10);
    cxt.lineTo(150, 50);
    cxt.lineTo(10, 50);
    cxt.lineTo(20, 10);
    cxt.stroke();
</script>
```

另类画圆

```
<canvas id="myCanvas" width="480" height="300" style="border:1px
solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>
<script type="text/javascript">
    var c = document.getElementById("myCanvas");
    var cxt = c.getContext("2d");
    var x = 150;
    var y = 150;
    var r = 100;
    cxt.moveTo(x - r, y);
    for (var i = x - r; i < x + r + 1; i++) {
        var tempY = Math.pow(r * r - (x - i) * (x - i), 1 / 2);
        cxt.lineTo(i, y + tempY);
    }
    cxt.moveTo(x - r, y);
    for (var i = x - r; i < x + r + 1; i++) {
        var tempY = Math.pow(r * r - (x - i) * (x - i), 1 / 2);
        cxt.lineTo(i, y - tempY);
    }
    cxt.stroke();
</script>
```

```
    <script src="jscex.min.js" type="text/javascript"></script>
```

```
    var somethingAsync = eval(Jscex.compile("async", function (a, b) {
        // implementation
    }));
```

```
<canvas id="myCanvas" width="480" height="300" style="border: 1px
solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>
    <script src="../Scripts/jscex.min.js" type="text/javascript"></script>
    <script type="text/javascript">
        var c = document.getElementById("myCanvas");
        var cxt = c.getContext("2d");
        var x = 150;
        var y = 150;
```

```
        var r = 100;
        var drawAsync = eval(Jscex.compile("async", function () {
            cxt.moveTo(x - r, y);
            for (var i = x - r; i < x + r + 1; i++) {
                $await(Jscex.Async.sleep(10));
                var tempY = Math.pow(r * r - (x - i) * (x - i), 1 / 2);
                cxt.lineTo(i, y + tempY);
                cxt.stroke();
            }
            cxt.moveTo(x - r, y);
            for (var i = x - r; i < x + r + 1; i++) {
                $await(Jscex.Async.sleep(10));
                var tempY = Math.pow(r * r - (x - i) * (x - i), 1 / 2);
                cxt.lineTo(i, y - tempY);
                cxt.stroke();
            }
        }));
        drawAsync().start();
    </script>
```

Sleep 一会儿

Sleep 一会儿

实验 4　超越祖冲之

简介

```
    <canvas id="myCanvas" width="480" height="300" style="border:
1px solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>
    <script type="text/javascript">
        var c = document.getElementById("myCanvas");
        var cxt = c.getContext("2d");
        var x = 150;
        var y = 150;
        var r = 100;
        var girth = 0;



        for (var i = x - r; i < x + r + 1; i += 1) {
```

这里代表 *x* 坐标

```
            var tempY1 = Math.pow(r * r - (x - i) * (x - i), 1 / 2);
            cxt.lineTo(i, y + tempY1);
            var tempX = r * r - (x - i + 1) * (x - i + 1);

            var tempY2 = Math.pow(r * r - (x - i + 1) * (x - i + 1), 1 / 2);
            var temp = getLengthOfTwoPoint({ x: i - 1, y: tempY2 }, { x: i,
y: tempY1 });
            if (tempX >= 0) {
                girth += temp;
            }
        }
        alert("π的是值为：" + girth / r)

        function getLengthOfTwoPoint(startPoint, endPoint) {
            var calX = Math.abs(startPoint.x - endPoint.x);
            var calY = Math.abs(startPoint.y - endPoint.y);
            return Math.pow((calX * calX + calY * calY), 0.5);
        }
    </script>
```

这里+1 代表下一个 x 坐标

用于求两点之间的距离

```
        var c = document.getElementById("myCanvas");
        var cxt = c.getContext("2d");
        var x = 150;
        var y = 150;
        var r = 400000;
        var girth = 0;
        for (var i = x - r; i < x + r + 1; i += 1) {
            var tempY1 = Math.pow(r * r - (x - i) * (x - i), 1 / 2);
            cxt.lineTo(i, y + tempY1);
            var tempX = r * r - (x - i + 1) * (x - i + 1);
            var tempY2 = Math.pow(r * r - (x - i + 1) * (x - i + 1), 1 / 2);
            var temp = getLengthOfTwoPoint({ x: i - 1, y: tempY2 }, { x: i,
y: tempY1 });
            if (tempX >= 0) {
                girth += temp;
            }
        }
        alert("π的是值为：" + girth / r)
```

```
        function getLengthOfTwoPoint(startPoint, endPoint) {
            var calX = Math.abs(startPoint.x - endPoint.x);
            var calY = Math.abs(startPoint.y - endPoint.y);
            return Math.pow((calX * calX + calY * calY), 0.5);
        }
```

实验 5　立 体 文 字

```
<canvas id="myCanvas" width="480" height="300" style="border: 1px
solid #c3c3c3;">
Your browser does not support the canvas element.
</canvas>
    <script>
        function draw() {
            var c = document.getElementById("myCanvas");
            var cxt = c.getContext("2d");
            cxt.fillStyle = "#0f0";
            var fontSize = 90;
            cxt.font = "bold 90px Arial";
            cxt.fillText("立", 190, 90);

            cxt.globalAlpha = 0.7;
            cxt.font = "bold 70px Arial";
            cxt.fillText("体", 260, 90);
            cxt.globalAlpha = 0.6;
            cxt.font = "bold 50px Arial";
            cxt.fillText("感", 310, 90);
            cxt.globalAlpha = 0.5;
            cxt.font = "bold 30px Arial";
            cxt.fillText("文", 350, 90);
            cxt.globalAlpha = 0.4;
            cxt.font = "bold 20px Arial";
            cxt.fillText("字", 370, 90);

            cxt.globalAlpha = 0.7;
            cxt.font = "bold 70px Arial";
            cxt.fillText("体", 145, 90);
            cxt.globalAlpha = 0.6;
            cxt.font = "bold 50px Arial";
```

以下为“立”右边的字

以下为“立”左边的字

```
            cxt.fillText("感", 110, 90);
            cxt.globalAlpha = 0.5;
            cxt.font = "bold 30px Arial";
            cxt.fillText("文", 90, 90);
            cxt.globalAlpha = 0.4;
            cxt.font = "bold 20px Arial";
            cxt.fillText("字", 80, 90);
        }
        draw();
    </script>
```

实验 6　鸟　　巢

椭圆绘制

```
        function drawEllipse(x, y, w, h) {
            //code here
        }
```

```
        for (var i = 0; i < points.length - 1; i++) {
            var p1 = points[i];
            var p2 = points[i + 1];
            context.beginPath();
            context.moveTo(p1.x, p1.y);
            context.lineTo(p2.x, p2.y);
            context.stroke();
        }
```

```
        context.beginPath();
        for (var i = 0; i < points.length - 1; i++) {
            var p1 = points[i];
            var p2 = points[i + 1];
            context.moveTo(p1.x, p1.y);
            context.lineTo(p2.x, p2.y);
        }
        context.stroke();
```

```
function drawEllipse(x, y, w, h) {
    var k = 0.55228475;
    var ox = (w / 2) * k;
    var oy = (h / 2) * k;
    var xe = x + w;
    var ye = y + h;
    var xm = x + w / 2;
    var ym = y + h / 2;
    ctx.beginPath();
    ctx.moveTo(x, ym);
    ctx.bezierCurveTo(x, ym - oy, xm - ox, y, xm, y);
    ctx.bezierCurveTo(xm + ox, y, xe, ym - oy, xe, ym);
    ctx.bezierCurveTo(xe, ym + oy, xm + ox, ye, xm, ye);
    ctx.bezierCurveTo(xm - ox, ye, x, ym + oy, x, ym);
    ctx.stroke();
}
```

旋转椭圆

```
var canvas;
var ctx;
var px = 0;
var py = 0;
function init() {
    canvas = document.getElementById("myCanvas2");
    ctx = canvas.getContext("2d");
    ctx.strokeStyle = "#fff";
    ctx.translate(70, 70);
}

init();
var i = 0;
function drawEllipse(x, y, w, h) {
    var k = 0.5522848;
    var ox = (w / 2) * k;
    var oy = (h / 2) * k;
    var xe = x + w;
    var ye = y + h;
    var xm = x + w / 2;
    var ym = y + h / 2;
```

```
            ctx.beginPath();
            ctx.moveTo(x, ym);
            ctx.bezierCurveTo(x, ym - oy, xm - ox, y, xm, y);
            ctx.bezierCurveTo(xm + ox, y, xe, ym - oy, xe, ym);
            ctx.bezierCurveTo(xe, ym + oy, xm + ox, ye, xm, ye);
            ctx.bezierCurveTo(xm - ox, ye, x, ym + oy, x, ym);
            ctx.stroke();
            ctx.translate(x + 70, y + 100);
            px = -70;
            py = -100;
            ctx.rotate(10 * Math.PI * 2 / 360);
        }

        var ct;
        var drawAsync = eval(Jscex.compile("async", function (ct) {
            while (true) {
                drawEllipse(px, py, 140, 200)
                $await(Jscex.Async.sleep(200, ct));
            }
        }))

        function start() {
            ct = new Jscex.Async.CancellationToken();
            drawAsync(ct).start();
        }

        function stop() {
            ct.cancel();
        }
```

```
    var xxxAsync = eval(Jscex.compile("async", function () {
    while (condition) {
        ....
        dosomething
        ....
        $await(Jscex.Async.sleep(1000));
    }
}))
```

```
        var xxxAsync = eval(Jscex.compile("async", function () {
            while (true) {
                if (condition) {
```

```
                //dosomething
                break;
            }
            //dosomething
            $await(Jscex.Async.sleep(1000));
        }
    }))
```

```
    var xxxAsync = eval(Jscex.compile("async", function () {
        while (true) {
            for (i in XXX) {
                if (condition) {
                    //要在这里跳出最外层的循环
                }
            }
            //dosomething
            $await(Jscex.Async.sleep(1000));
        }
    }))
```

```
    var xxxAsync = eval(Jscex.compile("async", function () {
        while (true) {
            for (i in XXX) {
                if (condition) {
                    //要在这里跳出最外层的循环
                    breakTag = true;
                }
            }
            if (breakTag) break;
            //dosomething
            $await(Jscex.Async.sleep(1000));
        }
    }))
```

```
    var countAsync1 = eval(Jscex.compile("async", function () {
        while (true) {
            for (i in XXX) {
                if (condition) {
                    //要在这里跳出最外层的循环
                }
```

```
            }
            $await(Jscex.Async.sleep(1000));
        }
    }))
    var countAsync2 = eval(Jscex.compile("async", function () {
        while (true) {
            for (i in XXX) {
                if (condition) {
                    //要在这里跳出最外层的循环
                }
            }
            $await(Jscex.Async.sleep(1000));
        }
    }))

    var executeAsyncQueue = eval(Jscex.compile("async", function () {
        while (true) {
            $await(countAsync1())
            $await(countAsync2())
            $await(Jscex.Async.sleep(1000));
        }

    }))
    executeAsyncQueue().start();
```

```
var executeAsyncQueue = eval(Jscex.compile("async", function () {
        while (true) {
            $await(countAsync1())
            $await(countAsync2())
            if (breakTag) break;
            $await(Jscex.Async.sleep(1000));
        }
    }))
```

```
    var xxAsync = eval(Jscex.compile("async", function () {
        while (true) {
            //dosomething
            $await(xxxAsync())
            if (breakTag) break;
            $await(Jscex.Async.sleep("1000"));
        }
```

```
    }))

    var xxxAsync = eval(Jscex.compile("async", function () {
        if (condition) {
            breakTag = true;
        }
        $await(Jscex.Async.sleep("1000"));
    }))
```

实验 7　贪　吃　蛇

贪吃蛇算法

```
var moveAsync = eval(Jscex.compile("async", function (p) {
    while (true) {
        //...
        //code here
        //...
        $await(Jscex.Async.sleep(100));
    }
}));
```

```
        if (direction == "right") {
            p.push({
                x: p[p.length - 1].x + 1,
                y: p[p.length - 1].y
            });
        }
        if (direction == "left") {
            p.push({
                x: p[p.length - 1].x - 1,
                y: p[p.length - 1].y
            });
        }
        if (direction == "up") {

            p.push({
                x: p[p.length - 1].x,
```

```
                    y: p[p.length - 1].y - 1
                });
            }
            if (direction == "down") {
                p.push({
                    x: p[p.length - 1].x,
                    y: p[p.length - 1].y + 1
                });
            }
```

```
            if (p[p.length - 1].x == foodPositions.x && p[p.length - 1].y ==
foodPositions.y) {
                foodPositions.x = -1;
                foodPositions.y = -1;
            }
            else {
                clearColor(p[0]);
                p.shift();
            }
```

```
   var MR = Math.random;

    function drawSnakeAndFood(p) {
        if (foodPositions.x == -1 && foodPositions.y == -1) {
            foodPositions.x = MR() * 40 | 0;
            foodPositions.y = MR() * 40 | 0;
        }
        cxt.fillStyle = randomColor();
        cxt.fillRect(foodPositions.x * width, foodPositions.y * height, height,
width);
        for (i = 0; i < p.length; i++) {
            cxt.fillStyle = "#D1EEEE";
            cxt.fillRect(p[i].x * width, p[i].y * height, height, width);
        }

    }
```

```
            if (p[p.length - 1].x < 0 || p[p.length - 1].x > 39 || p[p.length -
1].y < 0 || p[p.length - 1].y > 39) {
                gameOver = true;
```

```
            }

            if (isInSnake(p, { x: p[p.length - 1].x, y: p[p.length - 1].y })) {
                gameOver = true;
            }
```

```
    var moveAsync = eval(Jscex.compile("async", function (p) {
        while (true) {
            if (p[p.length - 1].x < 0 || p[p.length - 1].x > 39 || p[p.length -
1].y < 0 || p[p.length - 1].y > 39) {
                gameOver = true;
            }

            if (p[p.length - 1].x == foodPositions.x && p[p.length - 1].y ==
foodPositions.y) {
                foodPositions.x = -1;
                foodPositions.y = -1;
            }
            else {
                clearColor(p[0]);
                p.shift();
            }
            if (direction == "right") {
                p.push({
                    x: p[p.length - 1].x + 1,
                    y: p[p.length - 1].y
                });
            }
            if (direction == "left") {
                p.push({
                    x: p[p.length - 1].x - 1,
                    y: p[p.length - 1].y
                });
            }
            if (direction == "up") {

                p.push({
                    x: p[p.length - 1].x,
                    y: p[p.length - 1].y - 1
                });
            }
            if (direction == "down") {
```

```
            p.push({
                x: p[p.length - 1].x,
                y: p[p.length - 1].y + 1
            });
        }

        drawSnakeAndFood(p);
        if (gameOver) {
            drawGameOver();
            document.getElementById("btnReset").disabled = "";
            break;
        }

        $await(Jscex.Async.sleep(100));
    }
}));
```

第 2 章　物 理 实 验

实验 8　质点运动与反射

匀速直线运动

```
        var ball = {
            position: { x: 100, y: 100 },
            r: 15
        };
```

```
        var ball = {
            position: { x: 100, y: 100 },
            r: 15,
            vx: 190,
            vy: 110
        };
```

```
<script src="jscex.min.js" type="text/javascript"></script>
```

```
<canvas id="myCanvas" width="600" height="500">Your browser does
not support the canvas element.
 </canvas>
 <script type="text/javascript">
     var canvas = document.getElementById("myCanvas");
     var cxt = canvas.getContext("2d");
     var ball = {
         position: { x: 100, y: 100 },
         r: 15,
         vx: 190,
         vy: 110
     };
     var cyc = 10;
```

```
    var moveAsync = eval(Jscex.compile("async", function () {

        while (true) {
            cxt.fillStyle = "rgba(0, 0, 0, .3)";
            cxt.fillRect(0, 0, canvas.width, canvas.height);
            cxt.fillStyle = "#fff";
            cxt.beginPath();
            cxt.arc(ball.position.x, ball.position.y, ball.r, 0, Math.PI * 2,
true);
            cxt.closePath();
            cxt.fill();
            ball.position.x += ball.vx * cyc / 1000;
            ball.position.y += ball.vy * cyc / 1000;
            $await(Jscex.Async.sleep(cyc));
        }
    }))
    moveAsync().start();
</script>
```

实现了残影效果

小球的 x 坐标和 y 坐标每隔一个周期(cyc)就发生变化

```
        var moveAsync = eval(Jscex.compile("async", function () {
            while (true) {
                while (ball.position.x < 500) {
                    cxt.fillStyle = "rgba(0, 0, 0, .3)";
                    cxt.fillRect(0, 0, canvas.width, canvas.height);
                    cxt.fillStyle = "#fff";
                    cxt.beginPath();
                    cxt.arc(ball.position.x, ball.position.y, ball.r, 0, Math.PI
* 2, true);
                    cxt.closePath();
                    cxt.fill();
                    ball.position.x += ball.vx * cyc / 1000;
                    ball.position.y += ball.vy * cyc / 1000;
                    $await(Jscex.Async.sleep(cyc));
                }
                while (ball.position.x > 100) {
                    cxt.fillStyle = "rgba(0, 0, 0, .3)";
                    cxt.fillRect(0, 0, canvas.width, canvas.height);
                    cxt.fillStyle = "#fff";
                    cxt.beginPath();
                    cxt.arc(ball.position.x, ball.position.y, 15, 0, Math.PI *
2, true);
```

```
                    cxt.closePath();
                    cxt.fill();
                    ball.position.x -= ball.vx * cyc / 1000;
                    ball.position.y -= ball.vy * cyc / 1000;
                    $await(Jscex.Async.sleep(cyc));
                }
            }
        }))
        moveAsync().start();
```

```
        var canvas = document.getElementById("myCanvas");
        var cxt = canvas.getContext("2d");
        var ball = {
            x: 100,
            y: 100,
            r: 15,
            vx: 190,
            vy: 110
        };
        var cyc = 10;
        var moveAsync = eval(Jscex.compile("async", function () {
            while (true) {
                    cxt.fillStyle = "rgba(0, 0, 0, .3)";
                    cxt.fillRect(0, 0, canvas.width, canvas.height);
                    cxt.fillStyle = "#fff";
                    cxt.beginPath();
                    cxt.arc(ball.x, ball.y, ball.r, 0, Math.PI * 2, true);
                    cxt.closePath();
                    cxt.fill();
                    //与左右两壁碰撞
                    if (ball.r + ball.x > canvas.width || ball.x < ball.r)
ball.vx *= -1;
                    //与上下两壁碰撞
                    if (ball.r + ball.y > canvas.height || ball.y < ball.r)
ball.vy *= -1;
                    ball.x += ball.vx * cyc / 1000;
                    ball.y += ball.vy * cyc / 1000;
                    $await(Jscex.Async.sleep(cyc));
            }
        }))
        moveAsync().start();
```

实验 9　万 有 引 力

重力场下的永动机

```
var canvas = document.getElementById("myCanvas");
var cxt = canvas.getContext("2d");
var ball = {
    x: 100,
    y: 100,
    r: 15,
    vx: 0,
    vy: 0
};
var cyc = 10;
var a = 50;
var moveAsync = eval(Jscex.compile("async", function () {
    while (true) {
        cxt.fillStyle = "rgba(0, 0, 0, .3)";
        cxt.fillRect(0, 0, canvas.width, canvas.height);
        cxt.fillStyle = "#fff";
        cxt.beginPath();
        cxt.arc(ball.x, ball.y, ball.r, 0, Math.PI * 2, true);
        cxt.closePath();
        cxt.fill();
        ball.y += ball.vy * cyc / 1000;
        if (ball.r + ball.y >= canvas.height) {
            ball.vy *= -1;
        }
        else {
            ball.vy += a;
        }
        $await(Jscex.Async.sleep(cyc));
    }
}))
moveAsync().start();
```

位移是速度对时间的累积

速度是加速度对时间的累积

```
var canvas = document.getElementById("mycanvas");
var cxt = canvas.getContext("2d");
var ball = {
```

```
        x: 100,
        y: 100,
        r: 15,
        vx: 0,
        vy: 0
    };
    cxt.fillStyle = "#030303";
    cxt.fillRect(0, 0, canvas.width, canvas.height);
    cxt.fillStyle = "#fff";
    cxt.beginPath();
    cxt.arc(ball.x, ball.y, ball.r, 0, Math.PI * 2, true);
    cxt.closePath();
    cxt.fill();

    var hitCount = 0 ,cyc = 10, a = 50;
    var moveAsync2 = eval(Jscex.compile("async", function () {
        while (true) {
                cxt.fillStyle = "rgba(0, 0, 0, .3)";
                cxt.fillRect(0, 0, canvas.width, canvas.height);
                cxt.fillStyle = "#fff";
                cxt.beginPath();
                cxt.arc(ball.x, ball.y, ball.r, 0, Math.PI * 2, true);
                cxt.closePath();
                cxt.fill();
                ball.y += ball.vy * cyc / 1000;
                if (ball.r + ball.y >= canvas.height) {
                    if (ball.vy > 0) {
                        ball.vy *= -0.7;
                        hitCount++;
                    }
                }
                else {
                    ball.vy += a;
                }
                if (hitCount > 15) break;
                $await(Jscex.Async.sleep(cyc));
        }
    }))
```

碰撞之后，一部分动能转化为与地面碰撞产生的热能，所以速度降低，方向相反

实验 10　疯狂的大炮

平抛运动

```
        var balls = [];
        balls.length = 0;
        var ball1 = {
            x: 70,
            y: 50,
            r: 10,
            vx: 400,
            vy: 0
        };
        var ball2 = {
            x: 50,
            y: 50,
            r: 10,
            vx: 0,
            vy: 0
        };
        balls.push(ball1);
        balls.push(ball2);
```

```
        var canvas = document.getElementById("myCanvas");
        var cxt = canvas.getContext("2d");
        var cyc = 110;
        var a = 80;
        cxt.fillStyle = "#030303";
        cxt.fillRect(0, 0, canvas.width, canvas.height);
        var moveAsync = eval(Jscex.compile("async", function () {
            while (true) {
                cxt.fillStyle = "rgba(0, 0, 0, .3)";
                cxt.fillRect(0, 0, canvas.width, canvas.height);
                cxt.fillStyle = "#fff";
                for (var i in balls) {
                    cxt.beginPath();
                    cxt.arc(balls[i].x, balls[i].y, balls[i].r, 0, Math.PI * 2,
true);
                    cxt.closePath();
                    cxt.fill();
```

根据速度，重绘所有小球的位置

```
                    balls[i].y += balls[i].vy * cyc / 1000;
                    balls[i].x += balls[i].vx * cyc / 1000;
                    if (balls[i].r + balls[i].y >= canvas.height) {
                        if (balls[i].vy > 0) {
                            balls[i].vy *= -0.7;
                        }
                    }
                    else {
                        balls[i].vy += a;
                    }
                }
                $await(Jscex.Async.sleep(cyc));
            }
        }))
        moveAsync().start();
```

```
    var img = new Image();
    img.src = "image/artillery.png";
    img.onload = function () {
        cxt.drawImage(img, 0, 325);
    }
```

```
            var ball = {
                x: 185,
                y: 470,
                r: getRandomNumber(0, 20),
                vx: getRandomNumber(190, 3000),
                vy: getRandomNumber(-3000, 0)
            };
            balls.push(ball);
            if (balls.length > 100) {
                balls.shift();
            }
```

```
    function getRandomNumber(min, max) {
        return (min + Math.floor(Math.random() * (max - min + 1)))
    }
```

```
    var fireAsync = eval(Jscex.compile("async", function () {
```

```
        while (true) {
            var ball = {
                x: 185,
                y: 470,
                r: getRandomNumber(0, 20),
                vx: getRandomNumber(190, 3000),
                vy: getRandomNumber(-3000, 0)
            };
            balls.push(ball);
            if (balls.length > 100) {
                balls.shift();
            }
            cxt.fillStyle = "rgba(0, 0, 0, .3)";
            cxt.fillRect(0, 0, canvas.width, canvas.height);
            cxt.fillStyle = "#fff";
            cxt.drawImage(img, 0, 425);
            for (i in balls) {
                cxt.beginPath();
                cxt.arc(balls[i].x, balls[i].y, balls[i].r, 0, Math.PI * 2, true);
                cxt.closePath();
                cxt.fill();
                balls[i].y += balls[i].vy * cyc / 1000;
                balls[i].x += balls[i].vx * cyc / 1000;
                if (balls[i].r + balls[i].y >= canvas.height) {
                    if (balls[i].vy > 0) {
                        balls[i].vy *= -0.7;
                    }
                }
                else {
                    balls[i].vy += a;
                }
            }
            $await(Jscex.Async.sleep(cyc));
        }
    }))
    fireAsync().start();
```

实验 11　动能守恒不守恒你说了算

完全弹性碰撞

```
var canvas = document.getElementById("myCanvas");
var cxt = canvas.getContext("2d");
var r = 15;
var ball1 = {
    x: 20,
    y: 285,
    r: r,
    vx: 120,
    vy: 0
};
var ball2 = {
    x: 400,
    y: 285,
    r: r,
    vx: -30,
    vy: 0
};
var balls = [];
balls.push(ball1);
balls.push(ball2);
cxt.fillStyle = "#030303";
cxt.fillRect(0, 0, canvas.width, canvas.height);
function init() {
    cxt.fillStyle = "#fff";
    for (i in balls) {
        cxt.beginPath();
        cxt.arc(balls[i].x, balls[i].y, balls[i].r, 0, Math.PI * 2, true);
        cxt.closePath();
        cxt.fill();
    }
}
init();
var cyc = 20;
var moveAsync = eval(Jscex.compile("async", function () {
    while (true) {
        cxt.fillStyle = "rgba(0, 0, 0, .3)";
        cxt.fillRect(0, 0, canvas.width, canvas.height);
        cxt.fillStyle = "#fff";
```

假设两个小球处于同一水平面上，即 y 相同，并且一个沿 x 正方向运动，一个沿 x 负方向运动

```
                                                    碰撞检测，当两球的距
                                                    离小球 2*r 时发生碰撞
            if (Math.abs(ball1.x - ball2.x) <= 2 * r) {
                var temp = ball1.vx;
                ball1.vx = ball2.vx;
                ball2.vx = temp;
            }
            for (i in balls) {
                balls[i].x += balls[i].vx * cyc / 1000;
                cxt.beginPath();
                cxt.arc(balls[i].x, balls[i].y, balls[i].r, 0, Math.PI * 2, true);
                cxt.closePath();
                cxt.fill();
            }
            $await(Jscex.Async.sleep(cyc));
        }
    }))
```

非弹性碰撞

```
    var moveAsync = eval(Jscex.compile("async", function () {
        while (true) {
            cxt.fillStyle = "rgba(0, 0, 0, .3)";
            cxt.fillRect(0, 0, canvas.width, canvas.height);
            cxt.fillStyle = "#fff";
            if (Math.abs(ball1.x - ball2.x) <= 2 * r) {
                                                    动量总和，假设
                                                    质量都为 1
                var momentumSummation = ball1.vx + ball2.vx;
                var temp = ball2.vx;
                                                    假设损失了 65%的速度
                ball2.vx = ball1.vx * 0.65;
                                                    根据动量守恒，求出另外一个小球的速度
                ball1.vx = momentumSummation - ball2.vx;
```

```
            }
            for (i in balls) {
                balls[i].x += balls[i].vx * cyc / 1000;
                cxt.beginPath();
                cxt.arc(balls[i].x, balls[i].y, balls[i].r, 0, Math.PI * 2, true);
                cxt.closePath();
                cxt.fill();
            }
            $await(Jscex.Async.sleep(cyc));
        }
    }))
```

完全非弹性碰撞

```
    var moveAsync = eval(Jscex.compile("async", function () {
        while (true) {
            cxt.fillStyle = "rgba(0, 0, 0, .3)";
            cxt.fillRect(0, 0, canvas.width, canvas.height);
            cxt.fillStyle = "#fff";
            if (Math.abs(ball1.x - ball2.x) <= 2 * r) {
                v = (ball1.vx + ball2.vx) / 2;
                ball1.vx = v;
                ball2.vx = v;
            }
            for (i in balls) {
                balls[i].x += balls[i].vx * cyc / 1000;
                cxt.beginPath();
                cxt.arc(balls[i].x, balls[i].y, balls[i].r, 0, Math.PI * 2, true);
                cxt.closePath();
                cxt.fill();
            }
            $await(Jscex.Async.sleep(cyc));
        }
    }))
```

实验 12　密　闭　球

极坐标速度

```
var ball = {
    x: 100,
    y: 100,
    r: 15,
    vx: 100,
    vy: 100
};
```

```
 var ball = {
    x: 100,
    y: 100,
    r: 15,
    v: 100 * Math.sqrt(2),
    direction: Math.PI / 4
};
```

密闭球

```
    for (var i = 0; i < 6; i++) {
        var ball = {
            position: new Vector2(300, 250),
            r: 10,
            v: getRandomNumber(50, 200),
            direction: (getRandomNumber(0, 360) / 360) * 2 * Math.PI
        };
        balls.push(ball);
    }
```

```
    var bigCircularity = {
```

```
            position: new Vector2(300, 250),
            r: 190
        }
```

```
if (Math.round(Math.pow(balls[i].position.x - bigCircularity.position.x, 2) +
Math.pow(bigCircularity.position.y - balls[i].position.y, 2)) >=
Math.round(Math.pow(bigCircularity.r - balls[i].r, 2))) {

}
```

```
        var cD = balls[i].direction;
        var step = Math.acos((balls[i].position.x - bigCircularity.position.x) /
distance);
        if (balls[i].position.y > bigCircularity.position.y) {
            balls[i].direction = -cD - Math.PI + 2 * step;

        }else {
            balls[i].direction = -cD + Math.PI - 2 * step;
        }
```

```
         for (var i = 0; i < 1200; i++) {
            var ball = {
                position: new Vector2(300, 250),
                r: 1,
                v: getRandomNumber(50, 200),
                direction: (getRandomNumber(0, 360) / 360) * 2 * Math.PI
            };
            balls.push(ball);
        }
```

```
        for (var i = 0; i < 1200; i++) {
            var ball = {
                position: new Vector2(300, 150),
                r: 1,
                v: getRandomNumber(50, 200),
                direction: (getRandomNumber(0, 360) / 360) * 2 * Math.PI
            };
            balls.push(ball);
        }
```

实验 13　不规则的密室

简介

```
                    if (ball.r + ball.x > canvas.width || ball.x < ball.r)
ball.vx *= -1;
                    if (ball.r + ball.y > canvas.height || ball.y < ball.r)
ball.vy *= -1;
```

向量的优势

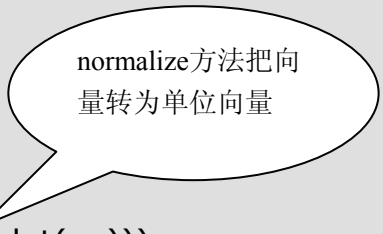
```
            function reflectionByLine(line) {
                var cp = line.getVerticalCrossoverPoint(this);
                this.reflectionByPoint(cp);
                return this;
            }
```

```
            function reflectionByPoint(point) {
                this.x = 2 * point.x - this.x;
                this.y = 2 * point.y - this.y;
                return this;
            }
```

```
            function getVerticalCrossoverPoint(v) {
                if (this.p2.x === this.p1.x) { return new Vector2(this.p1.x,
v.y); }
                if (this.p2.y === this.p1.y) { return new Vector2(v.x,
this.p1.y); }
                var k = (this.p2.y - this.p1.y) / (this.p2.x - this.p1.x);
                var cx = (k * k * this.p1.x + v.y * k + v.x - this.p1.y * k) /
(k * k + 1);
                var cy = k * cx - k * this.p1.x + this.p1.y;
                return new Vector2(cx, cy);
            }
```

```
Vector2 = function (x, y) {
```

```
    this.x = x || 0;
    this.y = y || 0;

};
Vector2.prototype = {
    constructor: Vector2,
    multiplyScalar: function (s) {
        this.x *= s;
        this.y *= s;
        return this;
    },
    divideScalar: function (s) {
        if (s) {
            this.x /= s;
            this.y /= s;
        } else {
            this.set(0, 0);
        }
        return this;
    },
    dot: function (v) {
        return this.x * v.x + this.y * v.y;
    },
    lengthSq: function () {
        return this.x * this.x + this.y * this.y;
    },
    length: function () {
        return Math.sqrt(this.lengthSq());
    },
    normalize: function () {
        return this.divideScalar(this.length());
    },
    reflectionSelf: function (v) {
        var nv = v.normalize();
        this.subSelf(nv.multiplyScalar(2 * this.dot(nv)));
    }
};
```




```
    var canvas = document.getElementById("mycanvas");
    var cxt = canvas.getContext("2d");
    cxt.fillStyle = "#030303";
```

```
    cxt.fillRect(0, 0, canvas.width, canvas.height);
    var balls = [];
    function getRandomNumber(min, max) {
        return (min + Math.floor(Math.random() * (max - min + 1)))
    }

    for (var i = 0; i < 100; i++) {
        var ball = {
            position: new Vector2(250, 200),
            r: getRandomNumber(6, 20),
            v:new Vector2( getRandomNumber(-200, 200),
getRandomNumber(-200, 200))
        };
        balls.push(ball);
    }

    var cyc = 10;
    var moveAsync = eval(Jscex.compile("async", function () {
        while (true) {
            cxt.fillStyle = "rgba(0, 0, 0, .3)";
            cxt.fillRect(0, 0, canvas.width, canvas.height);
            cxt.fillStyle = "#fff";
            for (i in balls) {
                cxt.beginPath();
                cxt.arc(balls[i].position.x, balls[i].position.y, balls[i].r, 0,
Math.PI * 2, true);
                cxt.closePath();
                cxt.fill();
                if (balls[i].r + balls[i].position.x > canvas.width ||
balls[i].position.x < balls[i].r) {

                    balls[i].v.reflectionSelf(new Vector2(1, 0));
                }
                if (balls[i].r + balls[i].position.y > canvas.height ||
balls[i].position.y < balls[i].r) {

                    balls[i].v.reflectionSelf(new Vector2(0, 1));
                }
                balls[i].position.x += balls[i].v.x * cyc / 1000;
                balls[i].position.y += balls[i].v.y * cyc / 1000;
            }
            $await(Jscex.Async.sleep(cyc));
```

产生 100 个速度随机、位置随机的小球

法向量为（1，0）

法向量为（0，1）

```
        }
    }))
```

```
        var p1 = new Vector2(0, 400);
        var p2 = new Vector2(300, 500);
        var p3 = new Vector2(600, 400);

        var p4 = new Vector2(0, 100);
        var p5 = new Vector2(300, 0);
        var p6 = new Vector2(600, 100);
```

```
                cxt.strokeStyle = "#fff";
                cxt.moveTo(p1.x, p1.y);
                cxt.lineTo(p2.x, p2.y);
                cxt.lineTo(p3.x, p3.y);
                cxt.moveTo(p4.x, p4.y);
                cxt.lineTo(p5.x, p5.y);
                cxt.lineTo(p6.x, p6.y);
                cxt.stroke();
```

```
                    if (balls[i].position.distanceToLine(p1, p2) < balls[i].r)
{
                        balls[i].v.reflectionSelf(Vector2.sub(p1,
p2).vertical());
                    }
                    if (balls[i].position.distanceToLine(p2, p3) < balls[i].r)
{
                        balls[i].v.reflectionSelf(Vector2.sub(p2,
p3).vertical());
                    }
                    if (balls[i].position.distanceToLine(p4, p5) < balls[i].r)
{
                        balls[i].v.reflectionSelf(Vector2.sub(p4,
p5).vertical());
                    }
                    if (balls[i].position.distanceToLine(p5, p6) < balls[i].r)
{
                        balls[i].v.reflectionSelf(Vector2.sub(p5,
p6).vertical());
                    }
```

```
distanceToLine: function (p1, p2) {
        if (p2.x === p1.x) {
            return Math.abs(this.y - p1.y);
        }
        else if (p2.y === p1.y) {
            return Math.abs(this.x - p1.x);
        }
        else {
            var A = (p2.y - p1.y) / (p2.x - p1.x);
            var B = -1;
            var C = p1.y - A * p1.x;
            return Math.abs(A * this.x + B * this.y + C) / Math.sqrt(A * A +
B * B);
        }
    },
```

```
    vertical: function () {
        return new Vector2(-this.y, this.x);
    }
```

```
    Vector2.sub(p1, p2).vertical()
```

实验 14　大球欺负小球

碰撞检测

```
         var l = bodyA.r + bodyB.r;
         var distSqr = bodyA.position.distanceToSquared(bodyB.position);
         var isTouching = distSqr <= l * l ? true : false;
```

```
    distanceToSquared: function (v) {
        var dx = this.x - v.x, dy = this.y - v.y;
```

```
        return dx * dx + dy * dy;
    },
```

```
    distanceTo: function (v) {
        return Math.sqrt(this.distanceToSquared(v));
    },
```

```
            var normal = Vector2.sub(bodyB.position,
bodyA.position).normalize();
            var ratio = bodyA.r / l;
            var contactPoint = new Vector2();
            contactPoint.x = bodyA.position.x + (bodyB.position.x -
bodyA.position.x) * ratio;
            contactPoint.y = bodyA.position.y + (bodyB.position.y -
bodyA.position.y) * ratio;
            var rA = Vector2.sub(contactPoint, bodyA.position);
            var rB = Vector2.sub(contactPoint, bodyB.position);
            var vrn = Vector2.sub(vA, vB).dot(normal);
```

```
Vector2.sub = function (v1, v2) {
    return new Vector2(v1.x - v2.x, v1.y - v2.y)
}
```

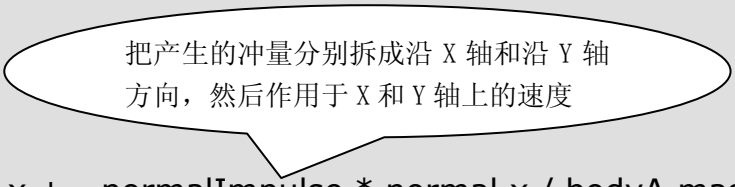
```
if (isTouching && vrn > 0) {

}
```

碰撞反应

```
        if (isTouching && vrn > 0) {
            var normalMass = 1 / (1 / bodyA.mass + 1 / bodyB.mass);
```

弹性系数取两球平均值

```
            var restitution = (bodyA.restitution + bodyB.restitution) / 2;
            var normalImpulse = -normalMass * vrn * (1 + restitution);
```




```
            bodyA.speed.x += normalImpulse * normal.x / bodyA.mass;
            bodyA.speed.y += normalImpulse * normal.y / bodyA.mass;
            bodyB.speed.x -= normalImpulse * normal.x / bodyB.mass;
            bodyB.speed.y -= normalImpulse * normal.y / bodyB.mass;
        }
```

多物体碰撞检测策略

```
                for (var i = 0; i < balls.length; i++) {
                    for (var j =0; j < balls.length; j++) {
                        collisionSolver(balls[i], balls[j]);
                    }
                }
```

```
                for (var i = 0; i < balls.length; i++) {
                    for (var j = i + 1; j < balls.length; j++) {
                        collisionSolver(balls[i], balls[j]);
                    }
                }
```

完整的代码

```
    var canvas = document.getElementById("myCanvas");
    var cxt = canvas.getContext("2d");
    var balls = [];
    function getRandomNumber(min, max) {
        return (min + Math.floor(Math.random() * (max - min + 1)))
```

```
        }
        for (var i = 0; i <40; i++) {
            var ball = {
                position: new Vector2(getRandomNumber(20, 600),
getRandomNumber(20, 600)),
                r: getRandomNumber(6, 20),
                speed:new Vector2(getRandomNumber(-200,
200),getRandomNumber(-200, 200)),
                mass:1,
                restitution: 1
            };
            balls.push(ball);
        }


        var filterBalls = [];
        for (var i = 0; i < balls.length; i++) {
            var overlapCount = 0;
            for (var j = i + 1; j < balls.length; j++) {
                var distance =
balls[i].position.distanceTo(balls[j].position);
                if (distance <= balls[i].r + balls[j].r) {
                    overlapCount++;
                }
            }
            if (overlapCount === 0) {
                filterBalls.push(balls[i]);
            }
        }
        balls = filterBalls;
        cxt.fillStyle = "#030303";
        cxt.fillRect(0, 0, canvas.width, canvas.height);


        function init() {
            cxt.fillStyle = "#fff";
            for (i in balls) {
                cxt.beginPath();
                cxt.arc(balls[i].position.x, balls[i].position.y, balls[i].r, 0,
Math.PI * 2, true); cxt.closePath();
```

生成 40 个小球

质量都设置为 1

弹性系数都设置为 1

过滤重叠在一起的球

把生成的小球先画上去

```
                cxt.fill();
            }
        }
        init();

        var cyc = 20;
        var moveAsync = eval(Jscex.compile("async", function () {
            var tag = 0;
            while (true) {
                try {
                    cxt.fillStyle = "rgba(0, 0, 0, .3)";
                    cxt.fillRect(0, 0, canvas.width, canvas.height);
                    cxt.fillStyle = "#fff";
                    for (var i = 0; i < balls.length; i++) {
                        for (var j = i + 1; j < balls.length; j++) {
                            collisionSolver(balls[i], balls[j]);
                        }
                    }
                    for (i in balls) {
                        cxt.beginPath();
                        cxt.arc(balls[i].position.x, balls[i].position.y,
balls[i].r, 0, Math.PI * 2, true);
                        cxt.closePath();
                        cxt.fill();
                        if (balls[i].r + balls[i].position.x > canvas.width) {
                            balls[i].position.x = canvas.width - balls[i].r;
                            balls[i].speed.x *= -1;
                        } if (balls[i].position.x < balls[i].r) {
                            balls[i].position.x = balls[i].r;
                            balls[i].speed.x *= -1;
                        } if (balls[i].r + balls[i].position.y > canvas.height)
{
                            balls[i].position.y = canvas.height - balls[i].r;
                            balls[i].speed.y *= -1;
                        }
                        if (balls[i].position.y < balls[i].r) {
                            balls[i].position.y = balls[i].r;
                            balls[i].speed.y *= -1;
                        }

                        balls[i].position.x += balls[i].speed.x * cyc / 1000;
```

定义小球运动的任务

处理碰撞

位移变化等于对速度进行积分操作

```
                        balls[i].position.y += balls[i].speed.y* cyc / 1000;
                    }
                } catch (e) { alert(e); }
                $await(Jscex.Async.sleep(cyc));
            }
        }))

        function collisionSolver(bodyA, bodyB) {
            var vB = bodyB.speed;
            var l = bodyA.r + bodyB.r;
            var distSqr =
bodyA.position.distanceToSquared(bodyB.position);
            var isTouching = distSqr <= l * l ? true : false;
            var normal = Vector2.sub(bodyB.position,
bodyA.position).normalize();
            var ratio = bodyA.r / l;
            var contactPoint = new Vector2();
            contactPoint.x = bodyA.position.x + (bodyB.position.x -
bodyA.position.x) * ratio;
            contactPoint.y = bodyA.position.y + (bodyB.position.y -
bodyA.position.y) * ratio;

            var rA = Vector2.sub(contactPoint, bodyA.position);
            var rB = Vector2.sub(contactPoint, bodyB.position);
            var vrn = Vector2.sub(vA, vB).dot(normal);
            if (isTouching && vrn > 0) {
                var normalMass = 1 / (1 / bodyA.mass + 1 / bodyB.mass);
                var restitution = (bodyA.restitution + bodyB.restitution) /
2;
                var normalImpulse = -normalMass * vrn * (1 + restitution);
                bodyA.speed.x += normalImpulse * normal.x /
bodyA.mass;
                bodyA.speed.y += normalImpulse * normal.y /
bodyA.mass;
                bodyB.speed.x -= normalImpulse * normal.x /
bodyB.mass;
                bodyB.speed.y -= normalImpulse * normal.y /
bodyB.mass;
            }
        }
```

处理碰撞的函数

```
        function collisionSolver(bodyA, bodyB) {
            bodyA.mass = bodyA.r * 2;
            bodyB.mass = bodyB.r * 2;

            .......
            .......
            .......
         }
```

```
    r: getRandomNumber(6, 120),
```

第 3 章　3D 实 验

实验 15　立　方　体

投影分析

```
Vector3 = function (x, y, z) {
    this.x = x;
    this.y = y;
    this.z = z;
 };
```

```
        var Points = [];
        Points[0] = new Vector3(100, 100, 100);
        Points[1] = new Vector3(100, 100, -100);
        Points[2] = new Vector3(-100, 100, -100);
        Points[3] = new Vector3(-100, 100, 100);
        Points[4] = new Vector3(100, -100, 100);
        Points[5] = new Vector3(100, -100, -100);
        Points[6] = new Vector3(-100, -100, -100);
        Points[7] = new Vector3(-100, -100, 100);
```

```
        var c = document.getElementById("myCanvas");
        var cxt = c.getContext("2d");
        cxt.translate(250, 250);
        cxt.scale(1, -1);
```

```
        var distance = 500;
        var eyePosition = new Vector3(0, 0, 700)
```

```
        function projection() {
```

```
            for (var i = 0; i < Points.length; i++) {
                Points[i].x = Points[i].x * distance / Math.abs(eyePosition.z
- Points[i].z);
                Points[i].y = Points[i].y * distance / Math.abs(eyePosition.z
- Points[i].z);
            }
        }
        projection();
```

```
        function drawCube() {
            cxt.beginPath();
            cxt.moveTo(Points[0].x, Points[0].y);
            cxt.lineTo(Points[1].x, Points[1].y);
            cxt.lineTo(Points[2].x, Points[2].y);
            cxt.lineTo(Points[3].x, Points[3].y);
            cxt.lineTo(Points[0].x, Points[0].y);
            cxt.moveTo(Points[4].x, Points[4].y);
            cxt.lineTo(Points[5].x, Points[5].y);
            cxt.lineTo(Points[6].x, Points[6].y);
            cxt.lineTo(Points[7].x, Points[7].y);
            cxt.lineTo(Points[4].x, Points[4].y);
            cxt.moveTo(Points[1].x, Points[1].y);
            cxt.lineTo(Points[5].x, Points[5].y);
            cxt.moveTo(Points[0].x, Points[0].y);
            cxt.lineTo(Points[4].x, Points[4].y);
            cxt.moveTo(Points[2].x, Points[2].y);
            cxt.lineTo(Points[6].x, Points[6].y);
            cxt.moveTo(Points[3].x, Points[3].y);
            cxt.lineTo(Points[7].x, Points[7].y);
            cxt.stroke();
        }
        drawCube();
```

```
        var canvas = document.getElementById("myCanvas");
        var cxt = canvas.getContext("2d");
        cxt.lineWidth = 3;
        var Points = [];
        var translateX = 250;
        var translateY = 250;
        var distance = 500;
        var eyePosition = new Vector3(0, 0, 700);
```

```
        cxt.translate(translateX, translateY);
        cxt.scale(1, -1);
        function init() {
            Points[0] = new Vector3(100, 100, 100);
            Points[1] = new Vector3(100, 100, -100);
            Points[2] = new Vector3(-100, 100, -100);
            Points[3] = new Vector3(-100, 100, 100);
            Points[4] = new Vector3(100, -100, 100);
            Points[5] = new Vector3(100, -100, -100);
            Points[6] = new Vector3(-100, -100, -100);
            Points[7] = new Vector3(-100, -100, 100)
        }
        function changeDistance() {
            for (var i = 0; i < Points.length; i++) {
                Points[i].x = Points[i].x * distance / Math.abs(eyePosition.z
- Points[i].z);
                Points[i].y = Points[i].y * distance / Math.abs(eyePosition.z
- Points[i].z);
            }
        }
        function drawCube() {
            cxt.beginPath();
            cxt.moveTo(Points[0].x, Points[0].y);
            cxt.lineTo(Points[1].x, Points[1].y);
            cxt.lineTo(Points[2].x, Points[2].y);
            cxt.lineTo(Points[3].x, Points[3].y);
            cxt.lineTo(Points[0].x, Points[0].y);
            cxt.moveTo(Points[4].x, Points[4].y);
            cxt.lineTo(Points[5].x, Points[5].y);
            cxt.lineTo(Points[6].x, Points[6].y);
            cxt.lineTo(Points[7].x, Points[7].y);
            cxt.lineTo(Points[4].x, Points[4].y);
            cxt.moveTo(Points[1].x, Points[1].y);
            cxt.lineTo(Points[5].x, Points[5].y);
            cxt.moveTo(Points[0].x, Points[0].y);
            cxt.lineTo(Points[4].x, Points[4].y);
            cxt.moveTo(Points[2].x, Points[2].y);
            cxt.lineTo(Points[6].x, Points[6].y);
            cxt.moveTo(Points[3].x, Points[3].y);
            cxt.lineTo(Points[7].x, Points[7].y);
            cxt.stroke();
        }
        var reduceDrawCubeAsync = eval(Jscex.compile("async", function ()
{
```

```
                                        当摄像机到显示屏的距离大于
                                        750时，退出循环·

            while (distance < 750) {
                cxt.clearRect(-translateX, -translateY, canvas.width,
canvas.height);
                init();
                distance += 10;
                changeDistance();
                drawCube(10);
                $await(Jscex.Async.sleep(100));
            }
        }));
        var magnifyDrawCubeAsync = eval(Jscex.compile("async", function
() {
                                        当摄像机到显示屏的距离小于
                                        150时，退出循环·
            while (distance > 150) {
                cxt.clearRect(-translateX, -translateY, canvas.width,
canvas.height);
                init();
                distance -= 10;
                changeDistance();
                drawCube(-10);
                $await(Jscex.Async.sleep(100));
            }
        }));
        var executeAsync = eval(Jscex.compile("async", function () {
            $await(reduceDrawCubeAsync());
            $await(magnifyDrawCubeAsync());
        }));
        executeAsync().start();
```

3D 旋转

```
    function rotate(angle) {
        for (var i = 0; i < Points.length; i++) {
            var tempY = Points[i].y;
            Points[i].y = Points[i].y * Math.cos(angle) - Points[i].z *
```

```
Math.sin(angle);
            Points[i].z = tempY * Math.sin(angle) + Points[i].z *
Math.cos(angle);
        }
    }
```

```
    var canvas = document.getElementById("myCanvas4");
    var cxt = canvas.getContext("2d");
    cxt.lineWidth = 3;
    var Points = [];
    var translateX = 250;
    var translateY = 250;
    var distance3 = 500;
    var eyePosition4 = { x: 0, y: 0, z: 700 };
    cxt.translate(translateX, translateY);
    cxt.scale(1, -1);
    function init() {
        Points[0] = new Vector3(100, 100, 100);
        Points[1] = new Vector3(100, 100, -100);
        Points[2] = new Vector3(-100, 100, -100);
        Points[3] = new Vector3(-100, 100, 100);
        Points[4] = new Vector3(100, -100, 100);
        Points[5] = new Vector3(100, -100, -100);
        Points[6] = new Vector3(-100, -100, -100);
        Points[7] = new Vector3(-100, -100, 100)
    }
    function changedistance() {
        for (var i = 0; i < Points.length; i++) {
            Points[i].x = Points[i].x * distance3 / Math.abs(eyePosition4.z -
Points[i].z);
            Points[i].y = Points[i].y * distance3 / Math.abs(eyePosition4.z -
Points[i].z);
        }
    }
    var currentAngle = 0;
    var drawCube = function () {
        cxt.strokeStyle = randomColor();
        cxt.beginPath();
        cxt.moveTo(Points[0].x, Points[0].y);
        cxt.lineTo(Points[1].x, Points[1].y);
        cxt.lineTo(Points[2].x, Points[2].y);
        cxt.lineTo(Points[3].x, Points[3].y);
```

```
        cxt.lineTo(Points[0].x, Points[0].y);
        cxt.moveTo(Points[4].x, Points[4].y);
        cxt.lineTo(Points[5].x, Points[5].y);
        cxt.lineTo(Points[6].x, Points[6].y);
        cxt.lineTo(Points[7].x, Points[7].y);
        cxt.lineTo(Points[4].x, Points[4].y);
        cxt.moveTo(Points[1].x, Points[1].y);
        cxt.lineTo(Points[5].x, Points[5].y);
        cxt.moveTo(Points[0].x, Points[0].y);
        cxt.lineTo(Points[4].x, Points[4].y);
        cxt.moveTo(Points[2].x, Points[2].y);
        cxt.lineTo(Points[6].x, Points[6].y);
        cxt.moveTo(Points[3].x, Points[3].y);
        cxt.lineTo(Points[7].x, Points[7].y);
        cxt.stroke();
    }
    init();
    drawCube();
    var rotateAsync = eval(Jscex.compile("async", function () {
        while (true) {
            cxt.clearRect(-translateX, -translateY, canvas.width,
canvas.height);
            init();
            rotate(degToRad(currentAngle));
            currentAngle += 5;
            changedistance();
            drawCube();
            $await(Jscex.Async.sleep(100));
        }
    }));
    function degToRad(a) {
        return a*Math.PI/180;
    }
    function rotate(angle) {
        for (var i = 0; i < Points.length; i++) {
            var tempY = Points[i].y;
            Points[i].y = Points[i].y * Math.cos(angle) - Points[i].z *
Math.sin(angle);
            Points[i].z = tempY * Math.sin(angle) + Points[i].z *
Math.cos(angle);
        }
    }
    function randomColor() {
        var arrHex = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B",
```

角度转弧度

```
"C", "D", "E", "F"]; var strHex = "#";
        var index;
        for (var i = 0; i < 6; i++) {
            index = Math.round(Math.random() * 15);
            strHex += arrHex[index];
        }
        return strHex;
    }
    rotateAsync().start();
```

```
    function rotate(angle) {
        for (var i = 0; i < Points.length; i++) {
            var tempX = Points[i].x;
            var tempZ = Points[i].z;
            Points[i].x = Points[i].x * Math.cos(angle) - Points[i].z *
Math.sin(angle);
            Points[i].z = Points[i].z * Math.cos(angle) + tempX *
Math.sin(angle);
        }
    }
```

```
   function rotate(angle) {
        for (var i = 0; i < Points.length; i++) {
            var tempX = Points[i].x;
            var tempZ = Points[i].z;
            Points[i].x = Points[i].x * Math.cos(angle) - Points[i].z *
Math.sin(angle);
            Points[i].z = Points[i].z * Math.cos(angle) + tempX *
Math.sin(angle);
        }
        for (var i = 0; i < Points.length; i++) {
            var tempY = Points[i].y;
            Points[i].y = Points[i].y * Math.cos(angle) - Points[i].z *
Math.sin(angle);
            Points[i].z = tempY * Math.sin(angle) + Points[i].z *
Math.cos(angle);
        }
    }
```

实验 16　星 星 点 灯

球体模拟

```
    Vector3 = function (x, y, z) {
        this.x = x || 0;
        this.y = y || 0;
        this.z = z || 0;
    };
```

```
    var starPositions = [];
    var j = -1;
    for (var i = 0; i < 440; i++) {
        var xTemp = getRandomNumber(-250, 250);
        var yTemp = getRandomNumber(-250, 250);
        j *= -1;
        if (xTemp * xTemp + yTemp * yTemp <= r * r) {
            var zTemp = j * Math.sqrt(Math.abs(r * r - xTemp * xTemp -
yTemp * yTemp));
            starPositions.push(new Vector3(xTemp, yTemp, zTemp));
        }
    }

    function getRandomNumber(min, max) {
        return (min + Math.floor(Math.random() * (max - min + 1)))
    }
```

投影

```
    function positionsProjection() {
        for (var i = 0; i < starPositions.length; i++) {
            var tempV = projection(starPositions[i]);
```

```
            starPositionsForShow.push(tempV);
        }
    }
    function projection(v) {
        var v1 = new Vector3();
        v1.x = v.x * distance / Math.abs(cameraPosition.z - v.z);
        v1.y = v.y * distance / Math.abs(cameraPosition.z - v.z);
        v1.z = v.z;
        return v1;
    }
```

```
<script src="script/easel.js" type="text/javascript"></script>
```

```
    function init() {
        PositionsProjection();
        canvas = document.getElementById("testCanvas");
        stage = new Stage(canvas);
        for (var i = 0; i < starPositionsForShow.length; i++) {
            var star = new Bitmap("image/star.png");
            star.x = centreOfCirclePosition.x + starPositionsForShow[i].x;
            star.y = centreOfCirclePosition.x + starPositionsForShow[i].y;
            star.scaleX = star.scaleY = 0.5 * distance /
Math.abs(cameraPosition.z - starPositionsForShow[i].z);
            if (starPositionsForShow[i].z > 50) star.alpha = 1;
            if (starPositionsForShow[i].z < 50) star.alpha = 0.5;
            stage.addChild(star);
        }
        Ticker.setFPS(20);
        Ticker.addListener(window);
    }
```

```
function rotate(angle) {

        for (var i = 0; i < starPositions.length; i++) {
            var tempX = starPositions[i].x;
            var tempZ = starPositions[i].z;
            starPositions[i].x = starPositions[i].x * Math.cos(angle) -
starPositions[i].z * Math.sin(angle);
            starPositions[i].z = starPositions[i].z * Math.cos(angle) +
tempX * Math.sin(angle);
```

```
        }
    }
    function tick() {
        starPositionsForShow.length = 0;
        currentAngle += 0.0005;
        rotate(degToRad(currentAngle));
        PositionsProjection();

        for (var i = 0; i < starPositionsForShow.length; i++) {
            var star = stage.getChildAt(i);
            star.x = centreOfCirclePosition.x + starPositionsForShow[i].x;
            star.y = centreOfCirclePosition.x + starPositionsForShow[i].y;
            star.scaleX = star.scaleY = 0.5 * distance /
Math.abs(cameraPosition.z - starPositionsForShow[i].z);
            if (starPositionsForShow[i].z > 50) star.alpha = 1;
            if (starPositionsForShow[i].z < 50) star.alpha = 0.5;
            stage.addChild(star);
        }
        stage.update();
    }
    init();
```

实验 17　矩 阵 变 换

投影矩阵

```
    THREE.Matrix4.makeFrustum = function (left, right, bottom, top, near,
far) {

        var m, x, y, a, b, c, d;

        m = new THREE.Matrix4();

        x = 2 * near / (right - left);
        y = 2 * near / (top - bottom);

        a = (right + left) / (right - left);
        b = (top + bottom) / (top - bottom);
        c = -(far + near) / (far - near);
        d = -2 * far * near / (far - near);
```

```
        m.n11 = x; m.n12 = 0; m.n13 = a; m.n14 = 0;
        m.n21 = 0; m.n22 = y; m.n23 = b; m.n24 = 0;
        m.n31 = 0; m.n32 = 0; m.n33 = c; m.n34 = d;
        m.n41 = 0; m.n42 = 0; m.n43 = -1; m.n44 = 0;

        return m;

    };
```

```
Vector4 = function ( x, y, z, w ) {
    this.x = x || 0;
    this.y = y || 0;
    this.z = z || 0;
    this.w = w || 1;
};
```

实验模拟

```
      var Matrix4 = function (n11, n12, n13, n14, n21, n22, n23, n24, n31,
n32, n33, n34, n41, n42, n43, n44) {

          this.set(

        n11 || 1, n12 || 0, n13 || 0, n14 || 0,
        n21 || 0, n22 || 1, n23 || 0, n24 || 0,
        n31 || 0, n32 || 0, n33 || 1, n34 || 0,
        n41 || 0, n42 || 0, n43 || 0, n44 || 1

    );
      }

      Matrix4.prototype = {

          set: function (n11, n12, n13, n14, n21, n22, n23, n24, n31, n32,
n33, n34, n41, n42, n43, n44) {

              this.n11 = n11; this.n12 = n12; this.n13 = n13; this.n14 =
n14;
              this.n21 = n21; this.n22 = n22; this.n23 = n23; this.n24 =
```

```
n24;
                this.n31 = n31; this.n32 = n32; this.n33 = n33; this.n34 =
n34;
                this.n41 = n41; this.n42 = n42; this.n43 = n43; this.n44 =
n44;

                return this;
            },
            multiplyVector4: function (v) {

                var vx = v.x, vy = v.y, vz = v.z, vw = v.w;

                v.x = this.n11 * vx + this.n21 * vy + this.n31 * vz + this.n41
* vw;
                v.y = this.n12 * vx + this.n22 * vy + this.n32 * vz + this.n42
* vw;
                v.z = this.n13 * vx + this.n23 * vy + this.n33 * vz + this.n43
* vw;
                v.w = this.n14 * vx + this.n24 * vy + this.n34 * vz + this.n44
* vw;

                return v;

            }
        };
```

```
Vector4 = function ( x, y, z, w ) {

    this.x = x || 0;
    this.y = y || 0;
    this.z = z || 0;
    this.w = w || 1;

};
```

```
        var Points = [];
        function init() {
            Points[0] = new Vector4(100, 100, 100, 1);
            Points[1] = new Vector4(100, 100, -100, 1);
            Points[2] = new Vector4(-100, 100, -100, 1);
            Points[3] = new Vector4(-100, 100, 100, 1);
```

```
        Points[4] = new Vector4(100, -100, 100, 1);
        Points[5] = new Vector4(100, -100, -100, 1);
        Points[6] = new Vector4(-100, -100, -100, 1);
        Points[7] = new Vector4(-100, -100, 100, 1);
    }
```

```
    var m4 = new Matrix4();
    var angle = 0;
    var currentAngle = 0;
    function transform() {
        angle = degToRad(currentAngle)
        init();
        m4.n11 = Math.cos(angle);
        m4.n13 = -Math.sin(angle);
        m4.n31 = Math.sin(angle);
        m4.n33 = Math.cos(angle);
        for (var i = 0; i < Points.length; i++) {
            Points[i] = m4.multiplyVector4(Points[i]);
        }
    }
```

```
    var m4 = new Matrix4();
    function transform() {
        init();
        m4.n41++;
        m4.n42++;
        m4.n43++;
        for (var i = 0; i < Points.length; i++) {
            Points[i] = m4.multiplyVector4(Points[i]);
        }
    }
```

```
    var m4 = new Matrix4();
    function transform() {
        init();
        m4.n21 += 0.01;
        for (var i = 0; i < Points.length; i++) {
            Points[i] = m4.multiplyVector4(Points[i]);
```

```
        }
    }
```

```
    var m4 = new Matrix4();
    function transform() {
        init();
        m4.n11 += 0.01;
        m4.n22 += 0.01;
        m4.n33 += 0.01;
        for (var i = 0; i < Points.length; i++) {
            Points[i] = m4.multiplyVector4(Points[i]);
        }
    }
```

实验 18　3D 变形金刚蝙蝠侠

绘制立方体

```
    var Points = [];
    function init() {
        Points[0] = new Vector3(-100, 100, -100);
        Points[1] = new Vector3(-100, -100, -100);
        Points[2] = new Vector3(-100, 100, -100);
        Points[3] = new Vector3(100, 100, -100);
        Points[4] = new Vector3(100, -100, -100);
        Points[5] = new Vector3(100, 100, -100);
        Points[6] = new Vector3(100, 100, 100);
        Points[7] = new Vector3(100, -100, 100);
        Points[8] = new Vector3(100, 100, 100);
        Points[9] = new Vector3(-100, 100, 100);
        Points[10] = new Vector3(-100, -100, 100);
        Points[11] = new Vector3(-100, 100, 100);
    }
```

```
    rotateXSelf: function (p, theta) {
        var v = new Vector3();
        v.sub(this, p);
        theta *= Math.PI / 180;
        var R = [[Math.cos(theta), -Math.sin(theta)], [Math.sin(theta),
Math.cos(theta)]];

        this.y = p.y + R[0][0] * v.y + R[0][1] * v.z;
        this.z = p.z + R[1][0] * v.y + R[1][1] * v.z;
    },
    rotateYSelf: function (p, theta) {
        var v = new Vector3();
        v.sub(this, p);
        theta *= Math.PI / 180;
        var R = [[Math.cos(theta), -Math.sin(theta)], [Math.sin(theta),
Math.cos(theta)]];
        this.x = p.x + R[0][0] * v.x + R[0][1] * v.z;
        this.z = p.z + R[1][0] * v.x + R[1][1] * v.z;
    },
    rotateZSelf: function (p, theta) {
        var v = new Vector3();
        v.sub(this, p);
        theta *= Math.PI / 180;
        var R = [[Math.cos(theta), -Math.sin(theta)], [Math.sin(theta),
Math.cos(theta)]];
        this.x = p.x + R[0][0] * v.x + R[0][1] * v.y;
        this.y = p.y + R[1][0] * v.x + R[1][1] * v.y;
    }
```

```
        var executeAsyncQueue = eval(Jscex.compile("async", function () {
            $await(Jscex.Async.sleep(2000));
            $await(varyAsync1());
            $await(varyAsync2());
            $await(varyAsync3());
            $await(flyAsync());
        }))
        executeAsyncQueue().start();
```

```
        var varyAsync1 = eval(Jscex.compile("async", function () {
                while (true) {
                    try {
```

```
                        if (rAngle >= 90) {
                            step *= -1;
                            rAngle = 0;
                            break;
                        }
                        cxt.clearRect(-350, -350, canvas.width,
canvas.height);

                        rAngle += 5;
                        Points[5].rotateZSelf(Points[4], -step);
                        Points[6].rotateZSelf(Points[7], -step);

                        Points[0].rotateZSelf(Points[1], step);
                        Points[11].rotateZSelf(Points[10], step);

                        Points[2].rotateXSelf(Points[1], -step);
                        Points[3].rotateXSelf(Points[4], -step);

                        Points[9].rotateXSelf(Points[10], step);
                        Points[8].rotateXSelf(Points[7], step);
                        projection();
                        drawCube();
                        $await(Jscex.Async.sleep(150));
                    }
                    catch (e) {
                        alert(e);
                    }
                }
        }))
```

```
        var varyAsync2 = eval(Jscex.compile("async", function () {

            while (true) {
                try {
                    if (rAngle >= 490) {
                        step *= -1;
                        rAngle = 0;
                        break;
                    }
                    cxt.clearRect(-350, -350, canvas.width,
canvas.height);
```

```
                    rAngle += 5;
                    Points[2].x++;
                    Points[3].x--;
                    Points[0].z+=2;
                    Points[5].z+=2;
                    projection();
                    drawCube();
                    $await(Jscex.Async.sleep(15));
                }
                catch (e) {
                    alert(e);
                }
            }


        }))
```

```
        var varyAsync3 = eval(Jscex.compile("async", function () {

            while (true) {
                try {
                    if (rAngle >= 290) {
                        step *= -1;
                        rAngle = 0;
                        break;
                    }
                    cxt.clearRect(-350, -350, canvas.width,
canvas.height);

                    rAngle += 5;
                    Points[8].z -= 3;
                    Points[9].z -= 3;
                    Points[8].x++;
                    Points[9].x--;
                    projection();
                    drawCube();
                    $await(Jscex.Async.sleep(15));
                }
                catch (e) {
                    alert(e);
                }
            }
```

```
        }))
```

```
        var flyAsync = eval(Jscex.compile("async", function () {
            while (true) {
                for (var i in Points) {
                    Points[i].z -= 15;
                }
                while (true) {
                    try {
                        if (rAngle >= 20) {
                            step *= -1;
                            rAngle = 0;
                            break;
                        }
                        cxt.clearRect(-350, -350, canvas.width,
canvas.height);

                        rAngle += 5;
                        Points[5].rotateZSelf(Points[4], -step);
                        Points[6].rotateZSelf(Points[7], -step);

                        Points[0].rotateZSelf(Points[1], step);
                        Points[11].rotateZSelf(Points[10], step);


                        Points[9].rotateXSelf(Points[10], 3);
                        Points[8].rotateXSelf(Points[7], 3);
                        projection();
                        drawCube();
                        $await(Jscex.Async.sleep(15));
                    }
                    catch (e) {
                        alert(e);
                    }
                }
            }
        }))
```

实验 17　世界上最简单的 3D 场景渲染

地面渲染

```
        Camera = function (position) {
            this.position = position;
        }
        var k = 0;
        var m = 200;
        var n = 1000;
        var c = new Camera(new Vector3(k, m, n));
```

```
        Camera = function (position) {
            this.position = position;
        }
        var k = 0;
        var m = 200;
        var n = 1000;
        var c = new Camera(new Vector3(k, m, n));
        var distance = 100;
        var canvas = document.querySelector('canvas');
        var ctx = canvas.getContext('2d');
        var w = canvas.width;
        var h = canvas.height;
        var imgdata = ctx.getImageData(0, 0, 200, 200);
        render();
        function render() {
            var i = 0;
            var pixels = imgdata.data;

            for (var y = 0; y < h; y++) {
                for (var x = 0; x < w; x++) {
                    var a = -100 + x;
                    var b = 200 - y;
                    var v = new Vector3(a, b, 0);
                    //与zox平面的交点
                    var cv = new Vector3(m * a / (m - b), 0, n * b / (b - m));
                     if (Math.abs(cv.x) < 100 && cv.z > -4400 && cv.z < 0)
{
                        pixels[i] = 111;
```

```
                        pixels[i + 1] = 111;
                        pixels[i + 2] = 255;
                        pixels[i + 3] = 255 * (4400 - Math.abs(cv.z)) /
4400;


                    }
                    i += 4;
                }
            }
            ctx.putImageData(imgdata, 0, 0);
        }
        function getRandomNumber(min, max) {
            return (min + Math.floor(Math.random() * (max - min + 1)))
        }
```

墙壁渲染

```
        function render() {
            var i = 0;
            var pixels = imgdata.data;

            for (var y = 0; y < h; y++) {
                for (var x = 0; x < w; x++) {
                    var a = -100 + x;
                    var b = 200 - y;
                    var v = new Vector3(a, b, 0);
                    //与zox平面的交点
                    var cv = new Vector3(m * a / (m - b), 0, n * b / (b - m));

                    //与x=100平面的交点
                    var cx100 = new Vector3(100, (m * a - 100 * m + 100
* b) / a, (a * n - 100 * n) / a);

                    var cxx100 = new Vector3(-100, (m * a + 100 * m - 100
* b) / a, (a * n + 100 * n) / a);
                    if (a < 0 && cxx100.y > 0 && cxx100.y < 150 &&
cxx100.z > -4400) {

                        pixels[i] = 151;
                        pixels[i + 1] = getRandomNumber(163, 255);
                        pixels[i + 2] = 5;
                        pixels[i + 3] = 255 * (4400 - Math.abs(cxx100.z)) /
4400;
```

```
                    }
                    //与x=-100平面的交点
                    else if (a > 0 && cx100.y > 0 && cx100.y < 150 &&
cx100.z > -4400) {

                        pixels[i] = 151;
                        pixels[i + 1] = getRandomNumber(163, 255);
                        pixels[i + 2] = 5;
                        pixels[i + 3] = 255 * (4400 - Math.abs(cx100.z)) /
4400;
                    }

                    else if (Math.abs(cv.x) < 100 && cv.z > -4400 && cv.z <
0) {

                        pixels[i] = 111;
                        pixels[i + 1] = 111;
                        pixels[i + 2] = 255;
                        pixels[i + 3] = 255 * (4400 - Math.abs(cv.z)) /
4400;


                    }
                    i += 4;
                }
            }
            ctx.putImageData(imgdata, 0, 0);
        }
```

第 4 章　综 合 实 验

实验 20　正 *N*边形变换

绘制正多边形

```
function drawRegular(borderCount, position, radius) {
    //....
    //code here
    //...
}
```

```
function drawRegular(borderCount, position, radius) {

    if (borderCount < 3) return;
    var intervalAngle = 360 / borderCount;
    var firstPoint = new Vector2(position.x, position.y - radius);
    cxt.beginPath();
    cxt.moveTo(firstPoint.x, firstPoint.y);
    for (var i = 0; i < borderCount; i++) {
        firstPoint.rotateSelf(position, intervalAngle);
        cxt.lineTo(firstPoint.x, firstPoint.y);
    }
    cxt.stroke();

}
```

```
rotateSelf: function (p, theta) {
    var v = new Vector2();
    v.sub(this, p);
    theta *= Math.PI / 180;
    var R = [[Math.cos(theta), -Math.sin(theta)], [Math.sin(theta),
```

```
Math.cos(theta)]];
        this.x = p.x + R[0][0] * v.x + R[0][1] * v.y;
        this.y = p.y + R[1][0] * v.x + R[1][1] * v.y;
    },
```

```
v.sub(this, p);
```

```
    var canvas = document.getElementById("myCanvas");
    var cxt = canvas.getContext("2d");
    drawRegular(5, new Vector2(100, 100), 100);
```

```
    var beginX = 20, cyc = 1000, borderCount = 3;
    var drawAsync = eval(Jscex.compile("async", function () {
        while (true) {
            beginX += 80;
            drawRegular(borderCount, new Vector2(beginX, 100), 40);
            borderCount++;
            $await(Jscex.Async.sleep(cyc));
        }
    }))
```

实验 21　动态加载文字

效果实现

```
    Word = function (text, fontSize, color, position) {
        this.text = text;
        this.fontSize = fontSize;
        this.color = color;
        this.position = position;
    }
```

```
var textL = new Word("L", 30, "#ffffff", new Vector2(40, 60));
var textO = new Word("o", 30, "#ffffff", new Vector2(60, 60));
var textA = new Word("a", 30, "#ffffff", new Vector2(80, 60));
var textD = new Word("d", 30, "#ffffff", new Vector2(100, 60));
var textI = new Word("i", 30, "#ffffff", new Vector2(120, 60));
var textN = new Word("n", 30, "#ffffff", new Vector2(140, 60));
var textG = new Word("g", 30, "#ffffff", new Vector2(160, 60));
var textPoint1 = new Word(".", 30, "#ffffff", new Vector2(180, 60));
var textPoint2 = new Word(".", 30, "#ffffff", new Vector2(200, 60));
var textPoint3 = new Word(".", 30, "#ffffff", new Vector2(220, 60));
```

```
var text = [];
text.push(textL);
text.push(textO);
text.push(textA);
text.push(textD);
text.push(textI);
text.push(textN);
text.push(textG);
text.push(textPoint1);
text.push(textPoint2);
text.push(textPoint3);
```

```
var c = document.getElementById("myCanvas");
var cxt = c.getContext("2d");
cxt.fillStyle = "#000";
function drawAllWords() {
    cxt.font = "bolder " + textL.fontSize + "px 宋体";
    cxt.fillText(textL.text, textL.position.x, textL.position.y);
    cxt.font = "bolder " + textO.fontSize + "px 宋体";
    cxt.fillText(textO.text, textO.position.x, textO.position.y);
    cxt.font = "bolder " + textA.fontSize + "px 宋体";
    cxt.fillText(textA.text, textA.position.x, textA.position.y);
    cxt.font = "bolder " + textD.fontSize + "px 宋体";
    cxt.fillText(textD.text, textD.position.x, textD.position.y);
    cxt.font = "bolder " + textI.fontSize + "px 宋体";
    cxt.fillText(textI.text, textI.position.x, textI.position.y);
    cxt.font = "bolder " + textN.fontSize + "px 宋体";
    cxt.fillText(textN.text, textN.position.x, textN.position.y);
    cxt.font = "bolder " + textG.fontSize + "px 宋体";
    cxt.fillText(textG.text, textG.position.x, textG.position.y);
```

```
        cxt.font = "bolder " + textPoint1.fontSize + "px 宋体";
        cxt.fillText(textPoint1.text, textPoint1.position.x,
textPoint1.position.y);
        cxt.font = "bolder " + textPoint2.fontSize + "px 宋体";
        cxt.fillText(textPoint2.text, textPoint2.position.x,
textPoint2.position.y);
        cxt.font = "bolder " + textPoint3.fontSize + "px 宋体";
        cxt.fillText(textPoint3.text, textPoint3.position.x,
textPoint3.position.y);
    }
```

```
    var currentMap = 0;
    var drawAsync = eval(Jscex.compile("async", function () {
        while (true) {
            cxt.clearRect(0, 0, c.width, c.height);
            drawAllWords();
            if (currentMap > 395) currentMap = 0;
            currentMap += 5;
            if (parseInt(currentMap / 40) <= text.length - 1) {
                text[parseInt(currentMap / 40)].fontSize = 60 - currentMap
% 40;

            }
            if (parseInt(currentMap / 40) + 1 <= text.length - 1) {

                text[parseInt(currentMap / 40) + 1].fontSize = currentMap
% 40 + 20;
            }
            $await(Jscex.Async.sleep(10));

        }
    }));
    drawAsync().start();
```

插件化

```
        Loading = function (text, fontSize, baseFontSize, color, position,
interval, font, bolder) {
```

```
            this.text = text;
            this.fontSize = fontSize;
            this.baseFontSize = baseFontSize;
            this.color = color;
            this.position = position;
            this.interval = interval;
            this.font = font;
            this.bolder = bolder;
            return this.init();
        }

        Loading.prototype.init = function () {
            var text = [];
            var _this = this;
            var words = _this.text.split("");
            for (i in words) {
                text.push({
                    "text": words[i],
                    "fontSize": _this.fontSize,
                    "baseFontSize": _this.baseFontSize,
                    "color": _this.color,
                    "position": new Vector2(_this.position.x + i *
_this.interval, _this.position.y),
                    "font": _this.font,
                    "bolder": _this.bolder
                });

            }
            return text;
        }

        Vector2 = function (x, y) {
            this.x = x || 0;
            this.y = y || 0;
        };


        var loading = new Loading("this is from loadingCanvas", 30, 30,
"#ffffff", new Vector2(20, 80), 15, "宋体", "bolder");
        var loadingCanvas = document.createElement('canvas');
        loadingCanvas.width = 420;
        loadingCanvas.height = 100;

        var cxt = loadingCanvas.getContext("2d");
```

```
        cxt.fillStyle = loading[0].color;
        function drawLoading() {
            for (i in loading) {
                cxt.font = loading[i].bolder + " " + loading[i].fontSize + "px
" + loading[i].font;
                cxt.fillText(loading[i].text, loading[i].position.x,
loading[i].position.y);
            }

        }
        var currentMap = 0;
        function changeFontSize() {
            if (currentMap > 1100) currentMap = 0;
            currentMap += 5;
            if (parseInt(currentMap / 40) <= loading.length - 1) {
                loading[parseInt(currentMap / 40)].fontSize = 2 *
loading[0].baseFontSize - currentMap % 40;
            }
            if (parseInt(currentMap / 40) + 1 <= loading.length - 1) {

                loading[parseInt(currentMap / 40) + 1].fontSize =
currentMap % 40 + loading[0].baseFontSize;
            }
        }
        function draw() {
            cxt.clearRect(0, 0, loadingCanvas.width,
loadingCanvas.height);
            drawLoading();
            changeFontSize();
        }
        setInterval(draw, 15);
```

```
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext('2d');
        context.font = "bolder 40px 宋体";
        context.fillStyle = "#ffffff";
        var gameLoadingAsync = eval(Jscex.compile("async", function () {
            while (true) {
                context.clearRect(0, 0, canvas.width, canvas.height);
                context.drawImage(loadingCanvas, 0, 0);
                context.fillText("this is game canvas!", 10, 38);
                $await(Jscex.Async.sleep(10));
```

```
            }
        }))
        gameLoadingAsync().start();
```

```
var loading = new Loading("正在努力加载中...", 30, 30, "#ffffff", new
Vector2(20, 80), 15, "宋体", "bolder");
```

实验 22　Loading 图片

技术分析

```
    rotateSelf: function (p, theta) {
        var v = new Vector2();
        v.sub(this, p);
        theta *= Math.PI / 180;
        var R = [[Math.cos(theta), -Math.sin(theta)], [Math.sin(theta),
Math.cos(theta)]];
        this.x = p.x + R[0][0] * v.x + R[0][1] * v.y;
        this.y = p.y + R[1][0] * v.x + R[1][1] * v.y;
    },
```

```
 var loadingPosition = new Vector2(200, 200);
var loadingRadius = 50;
var intervalAngle = 45;
var bigCircleRadius = 8;
//loadingPosition正上方
var bigCirclePosition = new Vector2(200, 150);
function drawLoading() {
    for (var i = 0; i < 11; i++) {
        cxt.beginPath();
        cxt.arc(bigCirclePosition.x, bigCirclePosition.y, bigCircleRadius, 0,
Math.PI * 2, true);
        cxt.closePath();
        cxt.fill();
        bigCircleRadius -= 1;
        bigCirclePosition.rotateSelf(loadingPosition, 30);
    }
```

```
}
```

```
var rotateAsync = eval(Jscex.compile("async", function () {
    while (true) {
        bigCircleRadius = 10;
        cxt.clearRect(0, 0, canvas.width, canvas.height);
        drawLoading();
        $await(Jscex.Async.sleep(cyc));
    }
}))
rotateAsync().start();
```

实验 23　繁花之上，又生繁花

粒子系统

```
        var balls = [];
        function getRandomNumber(min, max) {
            return (min + Math.floor(Math.random() * (max - min + 1)))
        }
        function initBall() {
            for (var i = 0; i < 500; i++) {
                var ball = {
                    position: new Vector2(300, 250),
                    r: 2,
                    v: getRandomNumber(3, 200),
                    direction: (getRandomNumber(0, 360) / 360) * 2 *
Math.PI
                };
                balls.push(ball);
            }

        }
        initBall();
```

```
        var moveAsync = eval(Jscex.compile("async", function () {
            var t = 0;
            while (true) {
                var ballColor = randomColor();
                while (true) {

                    try {
                        t += cyc;
                        ctx.globalAlpha = 1;
                        ctx.fillStyle = "rgba(0, 0, 0, .3)";
                        ctx.fillRect(0, 0, canvas2.width, canvas2.height);
                        ctx.fillStyle = ballColor;
                        if (t > 1111) {

                            t = 0;
                            balls.length = 0;
                            initBall();
                            ctx.fillStyle = "#000000";
                            ctx.fillRect(0, 0, canvas2.width,
canvas2.height);
                            break;
                        }
                        ctx.globalAlpha = (1111 - t) / 1111;
                        for (i in balls) {
                            ctx.beginPath();
                            ctx.arc(balls[i].position.x, balls[i].position.y,
balls[i].r, 0, Math.PI * 2, true);
                            ctx.closePath();
                            ctx.fill();

                            balls[i].position.x += balls[i].v *
Math.cos(balls[i].direction) * cyc / 1000;
                            balls[i].position.y += balls[i].v *
Math.sin(balls[i].direction) * cyc / 1000;
                        }
                    }
                    catch (e) {
                        alert(e)
                    }
                    $await(Jscex.Async.sleep(cyc));
                }
                $await(Jscex.Async.sleep(100));
            }
        }))
```

```
        function randomColor() {

            var arrHex = ["#FFFF00", "#ADFF2F", "#7CFC00", "#8DEEEE",
"#00FF00", "#EE6AA7", "#E0FFFF", "#FFFFFF"];
            var strHex = "#";
            var index = Math.round(Math.random() * 7);
                return   arrHex[index];


        }
```

实验 24　心

笛卡尔心形曲线

```
            var c = document.getElementById("myCanvas");
            var cxt = c.getContext("2d");
            cxt.lineWidth = 6;
            cxt.strokeStyle = "#FF0000";
            cxt.translate(250, 250);
            cxt.scale(1, -1);
            cxt.beginPath();
            var drawHeart = eval(Jscex.compile("async", function () {
                var a = 112;
                for (var i = -5; i < 360; i++) {
                    x = a * (1 - Math.sin(i * Math.PI / 180)) * Math.cos(i *
Math.PI / 180);
                    y = a * (1 - Math.sin(i * Math.PI / 180)) * Math.sin(i *
Math.PI / 180);
                    cxt.lineTo(x, y);
                    cxt.stroke();
                    $await(Jscex.Async.sleep(10))
                }
            }))
            drawHeart().start();
```

两个椭圆构成的心

```
        var $id = function (n) {
            return document.getElementById(n) || n;
        }
        window.addEventListener("load", draw, false);
        var con = $id("pad").getContext('2d');
        con.fillStyle = '#f00'
        con.translate(300, 100);
        function draw() {
            var r = 0, a = 20, start = 0, end = 0;
            con.rotate(Math.PI);
            for (var q = 0; q < 1000; q++) {
                start += Math.PI * 2 / 1000;
                end = start + Math.PI * 2 / 1000;
                r = a * Math.sqrt(225 / (17 - 16 * Math.sin(start) *
Math.sqrt(Math.cos(start) * Math.cos(start))))
                con.arc(0, 0, r, start, end, false);
            }
            con.fill();
        }
```

完美的心

```
         var c = document.getElementById("myCanvas");
          var cxt = c.getContext("2d");
          cxt.lineWidth = 16;
          var startX = 250;
          var startY = 250;
          function getHeartPoint(c) {
              var b = c / Math.PI;
              var a = 10 * (16 * Math.pow(Math.sin(b), 3));
              var d = 10 * (13 * Math.cos(b) - 5 * Math.cos(2 * b) - 2 *
Math.cos(3 * b) - Math.cos(4 * b));
              return new Array(a, d)
          }
          var drawHeart = eval(Jscex.compile("async", function () {
              for (var z = 10; z < 30; z += 0.2) {
                  cxt.strokeStyle = randomColor();
                  var h = getHeartPoint(z);
```

```
                cxt.lineTo(startX + h[0], startY - h[1]);
                cxt.stroke();
                $await(Jscex.Async.sleep(100))
            }
        }))
```

实验 25　烟 花 易 冷

技术实现

```
        var cyc = 10;
        var a = 20;
        var balls = [];
        var ball = {
            x: 640,
            y: 520,
            r: 28,
            vx: 0,
            vy: -1250
        };
        balls.push(ball);
        var fireAsync = eval(Jscex.compile("async", function () {
            while (ball.y > 220) {
                cxt.fillStyle = "rgba(0, 0, 0, .3)";
                cxt.fillRect(0, 0, canvas.width, canvas.height);
                cxt.fillStyle = "#fff";
                cxt.drawImage(img, 600, 540);
                cxt.globalAlpha = 1;

                for (i in balls) {
                    cxt.beginPath();
                    cxt.arc(balls[i].x, balls[i].y, balls[i].r, 0, Math.PI * 2,
true);
                    cxt.closePath();
                    cxt.fill();
                    balls[i].y += balls[i].vy * cyc / 1000;
```

```
                balls[i].x += balls[i].vx * cyc / 1000;
                if (balls[i].r + balls[i].y >= canvas.height) {
                    if (balls[i].vy > 0) {
                        balls[i].vy *= -0.7;
                    }
                }
                else {
                    balls[i].vy += a;
                }
            }
            $await(Jscex.Async.sleep(cyc));
        }
    }))
```

```
    var balls = [];
    for (var i = 0; i < 700; i++) {
        var ball2 = {
            position: new Vector2(630, 220),
            r: getRandomNumber(1, 4),
            vx: getRandomNumber(-200, 200),
            vy: getRandomNumber(-200, 200),
            color: randomColor()
        };
        balls.push(ball2);
    }


    var t = 0;
    var explodeAsync = eval(Jscex.compile("async", function () {
        while (true) {
            try {
                if (t > 1000) {
                    t = 0;
                    break;
                }
                t += cyc;
                cxt.fillStyle = "rgba(0, 0, 0, .3)";
                cxt.fillRect(0, 0, canvas.width, canvas.height);

                cxt.drawImage(img, 600, 540);
                for (i in balls) {
                    cxt.fillStyle = balls[i].color;
```

```
                        cxt.beginPath();
                        cxt.arc(balls[i].position.x, balls[i].position.y,
balls[i].r, 0, Math.PI * 2, true);
                        cxt.closePath();
                        cxt.fill();
                        //反射
                        if (balls[i].r + balls[i].position.x > canvas.width ||
balls[i].position.x < balls[i].r) balls[i].vx *= -1;
                        if (balls[i].r + balls[i].position.y > canvas.height ||
balls[i].position.y < balls[i].r) balls[i].vy *= -1;
                        balls[i].position.x += balls[i].vx * cyc / 1000;
                        balls[i].position.y += balls[i].vy * cyc / 1000;
                    }
                }
                catch (e) {
                    alert(e);
                }
                $await(Jscex.Async.sleep(cyc));
            }
        }))
```

```
        var heartR = 110;
        var time = 0;
        var dropAsync = eval(Jscex.compile("async", function () {

            while (true) {
                time += cyc;
                try {
                    cxt.fillStyle = "rgba(0, 0, 0, .3)";
                    cxt.fillRect(0, 0, canvas.width, canvas.height);

                    cxt.drawImage(img, 600, 540);

                    for (i in ball2s) {
                        cxt.fillStyle = ball2s[i].color;
                        cxt.beginPath();
                        cxt.arc(ball2s[i].position.x, ball2s[i].position.y,
ball2s[i].r, 0, Math.PI * 2, true);
                        cxt.closePath();
                        cxt.fill();
                        ball2s[i].position.y += ball2s[i].vy * cyc / 1000;
                        ball2s[i].position.x += ball2s[i].vx * cyc / 1000;
```

```
                        if (ball2s[i].r + ball2s[i].position.x > canvas.width)
{
                            //边界值处理，防止连续进入条件
                            ball2s[i].position.x = canvas.width - ball2s[i].r;
                            ball2s[i].vx *= -0.9;
                        }


                        if (ball2s[i].position.x < ball2s[i].r) {
                            //边界值处理，防止连续进入条件
                            ball2s[i].position.x = ball2s[i].r;
                            ball2s[i].vx *= -0.9;
                        }


                        if (ball2s[i].r + ball2s[i].position.y > canvas.height)
{
                            //边界值处理，防止连续进入条件
                            ball2s[i].position.y = canvas.height -
ball2s[i].r;
                            ball2s[i].vy *= -0.5;
                        }


                        if (ball2s[i].position.y < ball2s[i].r) {
                            //边界值处理，防止连续进入条件
                            ball2s[i].position.y = ball2s[i].r;
                            ball2s[i].vy *= -0.5;
                        }
                        var x = ball2s[i].position.x - 630;
                        var y = 220 - ball2s[i].position.y;
                        var result = ((x / heartR) * (x / heartR) + (y /
heartR) * (y / heartR) - 1) * ((x / heartR) * (x / heartR) + (y / heartR) * (y
/ heartR) - 1) * ((x / heartR) * (x / heartR) + (y / heartR) * (y / heartR) - 1)
- (x / heartR) * (x / heartR) * (y / heartR) * (y / heartR) * (y / heartR);

                        if (result > 0) {

                            ball2s[i].vy += 140;
                        }
                        else {
                            ball2s[i].vy = 0;
                            ball2s[i].vx = 0;
```

```
                    }

                }
            } catch (e) {
                alert(e)
            }
            if (time > 800) {
                time = 0;
                break;
            }
            $await(Jscex.Async.sleep(cyc));
        }
    }))
```

```
    var dTime = 0;
    var drawAsync = eval(Jscex.compile("async", function () {
        while (true) {
            dTime += 100;
            // Once window is loaded we set the configuration and the
default styles.
            CT.config({
                canvas: canvas,
                context: cxt,
                fontFamily: "Verdana",
                fontSize: "162px",
                fontColor: "#ff5e99",
                fontFamily: "Times new roman",
                fontWeight: "bold",
                fontStyle: "italic"
            });
            // The text we want to draw.
            var textI = 'I';

            // Draw it!
            CT.drawText({
                text: textI,
                x: 330,
                y: 270,
                boxWidth: 1555
            });
            // The text we want to draw.
```

```
            var textU = 'U';

            // Draw it!
            CT.drawText({
                text: textU,
                x: 800,
                y: 290,
                boxWidth: 1555
            });
            if (dTime > 800) {
                dTime = 0;
                break;
            }
            $await(Jscex.Async.sleep(100));
        }

    }))
```

```
    var qAsync = eval(Jscex.compile("async", function () {
        while (true) {
            //发射
            $await(fireAsync());
            //爆炸
            $await(explodeAsync());
            //心形以外的粒子下坠
            $await(dropAsync());
            //♥的两边画I 和  U
            $await(drawAsync());
            $await(Jscex.Async.sleep(1000));
        }
    }))
    qAsync().start();
```

心碎与封心

```
                    var result = ((x / heartR) * (x / heartR) + (y / heartR) * (y / heartR) - 1) * ((x / heartR) * (x / heartR) + (y / heartR) * (y / heartR) - 1) * ((x / heartR) * (x / heartR) + (y / heartR) * (y / heartR) - 1) - (x / heartR) * (x / heartR) * (y / heartR) * (y / heartR) * (y / heartR);
```

```
                    if (result > 0) {
                        balls[i].vy += 140;
                    }
                    else {
                        count++;
                    }
```

```
                    var result = ((x / heartR) * (x / heartR) + (y / heartR) * (y / heartR) - 1) * ((x / heartR) * (x / heartR) + (y / heartR) * (y / heartR) - 1) * ((x / heartR) * (x / heartR) + (y / heartR) * (y / heartR) - 1) - (x / heartR) * (x / heartR) * (y / heartR) * (y / heartR) * (y / heartR);

                    if (result < 0) {
                        cxt.beginPath();
                        cxt.arc(balls[i].position.x, balls[i].position.y, balls[i].r, 0, Math.PI * 2, true);
                        cxt.closePath();
                        cxt.fill();
                    }
                    else {
                        balls[i].vx *= -1;
                        balls[i].vy *= -1;
                    }
```

实验 26　波

实现

```
    var c = document.getElementById("myCanvas");
    var cxt = c.getContext("2d");
    var angel = 2 * Math.PI;
    var step = Math.PI / 10;
    function draw() {
        cxt.clearRect(0, 0, 1000, 1000);
        for (var i = 0; i < 600; i += 10) {
            cxt.fillStyle = randomColor();
```

```
            cxt.beginPath();
            angel -= step;
            cxt.arc(i, 100, 7, 0, Math.PI * 2, true);
            cxt.closePath();
            cxt.fill();
        }
    }
    draw();
    function randomColor() {
        var arrHex = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B",
"C", "D", "E", "F"]; var strHex = "#";
        var index;
        for (var i = 0; i < 6; i++) {
            index = Math.round(Math.random() * 15);
            strHex += arrHex[index];
        }
        return strHex;
    }
```

```
   var c = document.getElementById("myCanvas");
    var cxt = c.getContext("2d");
    var angel = 2 * Math.PI;
    var step = Math.PI / 10;
    function draw() {
        cxt.clearRect(0, 0, 1000, 1000);
        for (var i = 0; i < 600; i += 10) {
            cxt.fillStyle = randomColor();
            cxt.beginPath();
            angel -= step;
            cxt.arc(i, 150 + 100 * Math.cos(angel), 7, 0, Math.PI * 2, true);
            cxt.closePath();
            cxt.fill();
        }
    }
    draw();
```

```
    var c = document.getElementById("myCanvas");
    var cxt = c.getContext("2d");
    var angel = 2 * Math.PI;
    var step = Math.PI / 10;
    function draw() {
```

```
        cxt.clearRect(0, 0, 1000, 1000);
        for (var i = 0; i < 600; i += 10) {
            cxt.fillStyle = randomColor();
            cxt.beginPath();
            angel -= step;
            cxt.arc(i, 150 + 100 * Math.cos(angel), 7, 0, Math.PI * 2, true);
            cxt.closePath();
            cxt.fill();
        }
    }
    var step2 = 0.2;
    var waveAsync = eval(Jscex.compile("async", function () {
        while (true) {
            angel = 2 * Math.PI;
            angel -= step2;
            step2 += 0.1;
            $await(Jscex.Async.sleep(100))
            draw();
        }
    }))
    waveAsync().start();
```

实验 27　粒子计数器

粒子计数器的实现

```
var canvas = document.getElementById("myCanvas");
var cxt = canvas.getContext("2d");
cxt.fillStyle = "#FF0000";

for (var i = 0; i < 4; i++) {
    for (var j = 0; j < 7; j++) {
        cxt.arc(100 + i * 20, 100 + j * 20, 5, 0, Math.PI * 2, true);
    }
}
```

```
    var one = [];
    one.push({ x: 3, y: 0 });
    one.push({ x: 3, y: 1 });
    one.push({ x: 3, y: 2 });
    one.push({ x: 3, y: 3 });
    one.push({ x: 3, y: 4 });
    one.push({ x: 3, y: 5 });
    one.push({ x: 3, y: 6 });
```

```
    var two = [];
    two.push({ x: 0, y: 0 });
    two.push({ x: 1, y: 0 });
    two.push({ x: 2, y: 0 });
    two.push({ x: 3, y: 0 });
    two.push({ x: 3, y: 1 });
    two.push({ x: 3, y: 2 });
    two.push({ x: 3, y: 3 });
    two.push({ x: 2, y: 3 });
    two.push({ x: 1, y: 3 });
    two.push({ x: 0, y: 3 });
    two.push({ x: 0, y: 4 });
    two.push({ x: 0, y: 5 });
    two.push({ x: 0, y: 6 });
    two.push({ x: 1, y: 6 });
    two.push({ x: 2, y: 6 });
    two.push({ x: 3, y: 6 });
```

```
  if (num === 1) {
        for (i in one) {
            var result = JSLINQ(two).
            Count(function (item) { return item.x == one[i].x && item.y ==
one[i].y; });
            if (result == 0) {
                var ball = { x: 100 + one[i].x * 20, y: 100 + one[i].y * 20,
r: 8, vx: getRandomNumber(-250, 250), vy: getRandomNumber(-150,
150) };
                dropBall.push(ball);
            }
        }
```

```
    }
```

```
    var drawAsync = eval(Jscex.compile("async", function () {
        while (true) {
                if (dropBall.length > 100) dropBall.shift();
                cxt.clearRect(0, 0, canvas.width, canvas.height);
                cxt.beginPath();
                draw(count % 10);
                for (i in dropBall) {
                    cxt.arc(dropBall[i].x, dropBall[i].y, dropBall[i].r, 0,
Math.PI * 2, true);
                    dropBall[i].y += dropBall[i].vy * cyc / 1000;
                    dropBall[i].x += dropBall[i].vx * cyc / 1000;
                    if (dropBall[i].r + dropBall[i].y > canvas.height) {
                        dropBall[i].vy *= -0.7;
                        dropBall[i].y = canvas.height - dropBall[i].r;
                    }
                    dropBall[i].vy += a;
                }
                cxt.fill();
                t += cyc;
                if (t >= 1000) {
                    getBalls(count % 10);
                    t = 0;
                    count++;
                    draw(count % 10);
                }
            $await(Jscex.Async.sleep(cyc));
        }
    }))
```

```
                t += cyc;
                if (t >= 1000) {
                    getBalls(count % 10);
                    t = 0;
                    count++;
                    draw(count % 10);
                }
```

实验 28　时间之沙

核心函数

```
// 全局变量
int radius = 50.0;
int X, Y;
int nX, nY;
int delay = 16;
// 初始化设置
void setup(){
size( 200, 200 );
strokeWeight( 10 );
frameRate( 15 );
X = width / 2;
Y = width / 2;
nX = X;
nY = Y;
}
// 绘画函数(自动loop)
void draw(){
radius = radius + sin( frameCount / 4 );
// Track circle to new destination
X+=(nX-X)/delay;
Y+=(nY-Y)/delay;
// Fill canvas grey
background( 100 );
// Set fill-color to blue
fill( 0, 121, 184 );
// Set stroke-color white
stroke(255);
// Draw circle
ellipse( X, Y, radius, radius );
}
// 用户鼠标在 Canvas移动时产生的action
void mouseMoved(){
nX = mouseX;
nY = mouseY;
}
```

```
<!DOCTYPE html>
<html>
<head>
<body>
<script src="processing.js" type="text/javascript"></script>
<script type="application/processing">
void draw() {
size(200, 200);background(0); fill(80); noStroke(); ellipse(100, 100, 160,
160); stroke(255);
line(100, 100, cos( TWO_PI*second()/60- HALF_PI) * 70 + 100,
sin(TWO_PI*second()/60- HALF_PI) * 70 + 100);
line(100, 100, cos( TWO_PI*minute()/60- HALF_PI) * 60 + 100,
sin(TWO_PI*minute()/60- HALF_PI) * 60 + 100);
line(100, 100, cos(TWO_PI*(hour()%12)/12- HALF_PI) * 50 + 100,
sin(TWO_PI*(hour()%12)/12- HALF_PI) * 50 + 100);
}
</script>
<canvas>你的浏览器不支持HTML5,请使用谷歌、IE9或者火狐浏览器··</canvas>
</body>
</html>
```

实验 29　心　碎

特效实现

```
var tex = "心";
cxt.fillStyle = "rgba(0,0,0,1)";
cxt.fillRect(0, 0, 430, 400);
cxt.fillStyle = "rgba(255,255,255,1)"
cxt.font = "bolder 400px 宋体";
cxt.textBaseline = 'top';
cxt.fillText(tex, 20, 20);
```

```
for (y = 1; y < 400; y += 10) {
        for (x = 1; x < 400; x += 10) {
            imageData = cxt.getImageData(20 + x, 20 + y, 1, 1);
            if (imageData.data[0] > 170) {
                ps.push({ px: 20 + x, py: 20 + y });
            }
        }
    }
```

```
    for (i in ps) {
        var ball = {
            x: ps[i].px,
            y: ps[i].py,
            r: 2,
            vx: getRandomNumber(-10, 10),
            vy: getRandomNumber(0, 100)
        };
        balls.push(ball);

    }
    cxt.fillStyle = "#fff";
    for (i in balls) {
        cxt.beginPath();
        cxt.arc(balls[i].x, balls[i].y, balls[i].r, 0, Math.PI * 2, true);
        cxt.closePath();
        cxt.fill();

    }
```

```
    var dropAsync = eval(Jscex.compile("async", function () {

        while (true) {
            if (breakTag) {
                break;
            }
            cxt.fillStyle = "rgba(0, 0, 0, .3)";
            cxt.fillRect(0, 0, canvas.width, canvas.height);
            cxt.fillStyle = "#fff";
```

```
            for (i in balls) {
                cxt.beginPath();
                cxt.arc(balls[i].x, balls[i].y, balls[i].r, 0, Math.PI * 2, true);
                cxt.closePath();
                cxt.fill();
                balls[i].y += balls[i].vy * cyc / 1000;
                balls[i].x += balls[i].vx * cyc / 1000;
                if (balls[i].r + balls[i].y >= canvas.height) {
                    if (balls[i].vy > 0) {
                        balls[i].vy *= -0.7;
                    }
                }
                else {
                    balls[i].vy += a;
                }
            }
            $await(Jscex.Async.sleep(cyc));
        }
    }))
```

实验 30　Canvas 类库

DOM 事件过渡给 Canvas 中的元素对象

```
    p._enableMouseEvents = function() {
        var o = this;
        var evtTarget = window.addEventListener ? window : document;
        evtTarget.addEventListener("mouseup", function(e)
{ o._handleMouseUp(e); }, false);
        evtTarget.addEventListener("mousemove", function(e)
{ o._handleMouseMove(e); }, false);
        evtTarget.addEventListener("dblclick", function(e)
{ o._handleDoubleClick(e); }, false);
        if (this.canvas) { this.canvas.addEventListener("mousedown",
function(e) { o._handleMouseDown(e); }, false); }
    }
```

```
    p._handleMouseDown = function(e) {
        if (this.onMouseDown) {
            this.onMouseDown(new MouseEvent("onMouseDown",
this.mouseX, this.mouseY, this, e));
        }
        var target = this._getObjectsUnderPoint(this.mouseX, this.mouseY,
null, (this._mouseOverIntervalID ? 3 : 1));
        if (target) {
            if (target.onPress instanceof Function) {
                var evt = new MouseEvent("onPress", this.mouseX,
this.mouseY, target, e);
                target.onPress(evt);
                if (evt.onMouseMove || evt.onMouseUp)
{ this._activeMouseEvent = evt; }
            }
            this._activeMouseTarget = target;
        }
    }
```

缓存机制

```
    p.cache = function(x, y, width, height) {
        if (this.cacheCanvas == null) { this.cacheCanvas =
document.createElement("canvas"); }
        var ctx = this.cacheCanvas.getContext("2d");
        this.cacheCanvas.width = width;
        this.cacheCanvas.height = height;
        ctx.clearRect(0, 0, width+1, height+1);
        ctx.setTransform(1, 0, 0, 1, -x, -y);
        this.draw(ctx, true, this._matrix.reinitialize(1,0,0,1,-x,-y));
        this._cacheOffsetX = x;
        this._cacheOffsetY = y;
        this._applyFilters();
        this.cacheID++;
    }
```

集成循环

```
        protected override void Update(GameTime gameTime)
        {
            // Allows the game to exit
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
ButtonState.Pressed)
                this.Exit();

            // TODO: Add your update logic here

            base.Update(gameTime);
        }
```

```
    Ticker.setInterval = function(interval) {
        Ticker._lastTime = Ticker._getTime();
        Ticker._interval = interval;
        if (Ticker.timeoutID != null) { clearTimeout(Ticker.timeoutID); }
        if (Ticker.useRAF) {
            if (Ticker._rafActive) { return; }
            Ticker._rafActive = true;
            var f = window.requestAnimationFrame ||
window.webkitRequestAnimationFrame ||
window.mozRequestAnimationFrame ||
                        window.oRequestAnimationFrame ||
window.msRequestAnimationFrame;
            if (f) {
                f(Ticker._handleAF);
                return;
            }
        }
        if (Ticker._inited) { Ticker.timeoutID =
setTimeout(Ticker._handleTimeout, interval); }
    }
```

下篇

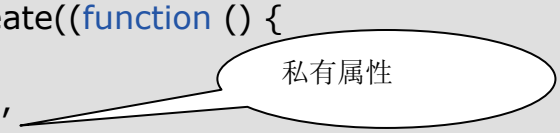
游 戏 开 发

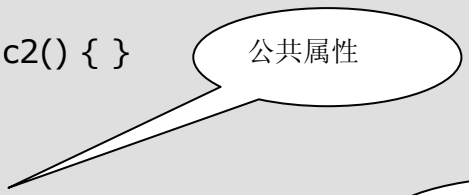
第 5 章　一步一步搭建物理引擎

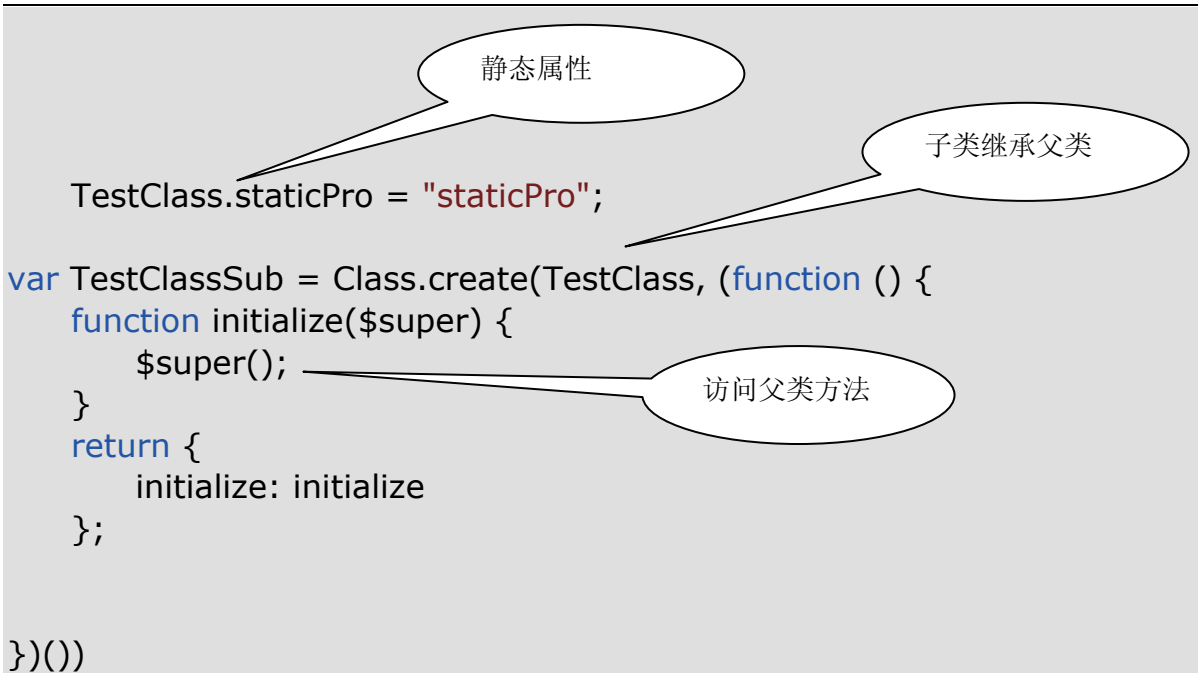
第 1 步　面向对象编程

Prototype.js

```
var TestClass = Class.create((function () {
        var
          _private_prop1,
          _private_prop2,


          public_prop;

        function _privateFunc1(a, b) { return a + b }

        function _privateFunc2() { }

        function publicStaticFunc1() {
            alert(_privateFunc1(1, 2));
        }

        function publicFunc2() { }

        return {

            public_prop: public_prop,

            publicStaticFunc1: publicStaticFunc1.setStatic(),

            publicFunc2: publicFunc2
        };

    } ()));
```








```
    TestClass.staticPro = "staticPro";

var TestClassSub = Class.create(TestClass, (function () {
    function initialize($super) {
        $super();
    }
    return {
        initialize: initialize
    };


})())
```

```
    var myNameSpase = {};
    myNameSpace.common = {};
    myNameSpace.common.test = function () { };
```

```
    var PE = PE || {};
```

第 2 步　建立基本对象

对象建立

```
PE.Body = Class.create((function () {
    function initialize(world) {
        //物理的ID,用来和图形化接口做mapping
        this.id = -1;
        //物理动态还是静态
        this.bodyType = PE.BodyType.Dynamic;
        //是否忽略重力
```

```
        this.ignoreGravity = true;
        //摩擦力
        this.friction = 0.2;
        //弹性系数
        this.restitution = 1.0;
        //物理所属的世界
        this.world = world;
        //位置
        this.position = PE.Vector2.Zero;
        //质量
        this.mass = 1;
        //速度
        this.speed = PE.Vector2.Zero;
        //施加在物体上的力
        this.force = PE.Vector2.Zero;
        //旋转的角度
        this.rotation = 0;
        //角速度
        this.palstance = 0;

        //把物体存入一个数组方便管理
        PE.Objects.push(this);
    }

    function integrateRotation(dt) {

        if (this.rotation >= 360) this.rotation %= 360;
        this.rotation += this.palstance * 180 * dt / Math.PI;
    }
    function integrateVelocity(dt) {
        this.speed.x += (this.force.x / this.mass) * dt;
        this.speed.y += (this.force.y / this.mass) * dt;
    }
    function integratePosition(dt) {
        this.position.x += this.speed.x * dt;
        this.position.y += this.speed.y * dt;

    }
    function clearForce() {
        this.force.x = 0;
        this.force.y = 0;
    }

    function getInvMass() {
```

```
        if (this.bodyType === PE.BodyType.Static) {
            return 0;
        }
        else {
            return 1 / this.mass;
        }

    }
    return {
        initialize: initialize,
        integrateVelocity: integrateVelocity,
        integratePosition: integratePosition,
        clearForce: clearForce,
        getInvMass: getInvMass,
        integrateRotation: integrateRotation
    };

} ()));
```

```
PE.BodyType = {
    Static: "Static",
    Dynamic: "Dynamic"
}
```

```
PE.World = Class.create((function () {
    function initialize(gravity) {
        PE.World.gravity = gravity;
    }
    function update(dt) {
        applyForces(dt);
    }
    function applyForces(dt) {
        for (var i = 0; i < PE.Objects.length; i++) {
            var obj = PE.Objects[i];
            var f = PE.Vector2.multiply(PE.World.gravity, obj.mass);
            obj.force.addSelf(f);
            obj.integrateVelocity(dt);
            obj.clearForce();
```

```
            obj.integrateRotation(dt);

            obj.integratePosition(dt);
        }
    }
    return {
        initialize: initialize,
        update: update.setStatic()
    };

} ()));

PE.World.gravity = new PE.Vector2(0, 9.8);
```

```
PE.Circle = Class.create(PE.Body, (function () {

    function initialize($super) {
        $super();
        this.radius = 10;
        this.type = "Circle";
    }
    function getRotationalInertia() {

        return this.mass * this.radius * this.radius;
    }

    function getInvRotationalInertia() {
        if (this.bodyType === PE.BodyType.Static) {
            return 0;
        }
        else {
            return 1 / (this.mass * this.radius * this.radius);
        }
    }
    return {
        initialize: initialize,
        getRotationalInertia: getRotationalInertia,
        getInvRotationalInertia: getInvRotationalInertia
    };
})())
```

```
PE.Rect = Class.create(PE.Body, (function () {
    function initialize($super) {
        $super();
        this.type = "Rect";

        this.width = 2;
        this.height = 2;
    }

    function getRotationalInertia() {

        return this.mass * (this.width * this.width + this.height *
this.height) / 12
    }

    function getInvRotationalInertia() {
        if (this.bodyType === PE.BodyType.Static) {
            return 0;
        }
        else {
            return 12 / (this.mass * (this.width * this.width + this.height *
this.height));
        }

    }
    return {
        initialize: initialize,
        getRotationalInertia: getRotationalInertia,
        getInvRotationalInertia: getInvRotationalInertia
    };

} ()));
```

```
PE.Vector2 = Class.create((function () {
    function initialize(x, y) {
        this.x = x || 0;
        this.y = y || 0;
    }
    function set(x, y) {
        this.x = x;
        this.y = y;
        return this;
```

```
}
function copy(v) {

    this.x = v.x;
    this.y = v.y;

    return this;

}
function clone() {

    return new PE.Vector2(this.x, this.y);

}

function add(v1, v2) {

    this.x = v1.x + v2.x;
    this.y = v1.y + v2.y;

    return this;

}

function addSelf(v) {

    this.x += v.x;
    this.y += v.y;

    return this;

}

function sub(v1, v2) {



    return new PE.Vector2(v1.x - v2.x, v1.y - v2.y);
}

function subSelf(v) {

    this.x -= v.x;
```

```
        this.y -= v.y;

        return this;

    }
    function multiply(v, s) {
        return new PE.Vector2(v.x * s, v.y * s);
    }
    function multiplyScalar(s) {

        this.x *= s;
        this.y *= s;

        return this;

    }

    function divideScalar(s) {

        if (s) {

            this.x /= s;
            this.y /= s;

        } else {

            this.set(0, 0);

        }

        return this;

    }


    function negate() {

        return this.multiplyScalar(-1);

    }

    function dot(v) {

        return this.x * v.x + this.y * v.y;
```

```
}

function lengthSq() {

    return this.x * this.x + this.y * this.y;

}

function length() {

    return Math.sqrt(this.lengthSq());

}

function normalize() {

    return this.divideScalar(this.length());

}

function distanceTo(v) {

    return Math.sqrt(this.distanceToSquared(v));

}

function distanceToSquared(v) {

    var dx = this.x - v.x, dy = this.y - v.y;
    return dx * dx + dy * dy;

}


function setLength(l) {

    return this.normalize().multiplyScalar(l);

}

function equals(v) {

    return ((v.x === this.x) && (v.y === this.y));
```

```
    }
    return {
        initialize: initialize,
        set: set,
        copy: copy,
        clone: clone,
        add: add.setStatic(),
        addSelf: addSelf,
        sub: sub.setStatic(),
        subSelf: subSelf,
        multiplyScalar: multiplyScalar,
        divideScalar: divideScalar,
        negate: negate,
        dot: dot,
        lengthSq: lengthSq,
        length: length,
        normalize: normalize,
        distanceTo: distanceTo,
        distanceToSquared: distanceToSquared,
        setLength: setLength,
        equals: equals

    };
} ()));

PE.Vector2.Zero = new PE.Vector2(0, 0);
```

第 3 步　集成单元测试框架

YUI Test Framework

```
YUI().use("node", "console", "test",function (Y) {

    Y.namespace("example.test");

    Y.example.test.DataTestCase = new Y.Test.Case({
```

```
        //name of the test case - if not provided, one is auto-generated
        name : "Data Tests",

        //---------------------------------------------------------------------
        // setUp and tearDown methods - optional
        //---------------------------------------------------------------------

        /*
         * Sets up data that is needed by each test.
         */
        setUp : function () {
            this.data = {
                name: "test",
                year: 2007,
                beta: true
            };
        },

        /*
         * Cleans up everything that was created by setUp().
         */
        tearDown : function () {
            delete this.data;
        },

        //---------------------------------------------------------------------
        // Test methods - names must begin with "test"
        //---------------------------------------------------------------------

        testName : function () {
            var Assert = Y.Assert;

            Assert.isObject(this.data);
            Assert.isString(this.data.name);
            Assert.areEqual("test", this.data.name);
        },

        testYear : function () {
            var Assert = Y.Assert;

            Assert.isObject(this.data);
            Assert.isNumber(this.data.year);
            Assert.areEqual(2007, this.data.year);
        },
```

```
    testBeta : function () {
        var Assert = Y.Assert;

        Assert.isObject(this.data);
        Assert.isBoolean(this.data.beta);
        Assert.isTrue(this.data.beta);
    }

});

Y.example.test.ArrayTestCase = new Y.Test.Case({

    //name of the test case - if not provided, one is auto-generated
    name : "Array Tests",

    //---------------------------------------------------------------------
    // setUp and tearDown methods - optional
    //---------------------------------------------------------------------

    /*
     * Sets up data that is needed by each test.
     */
    setUp : function () {
        this.data = [0,1,2,3,4]
    },

    /*
     * Cleans up everything that was created by setUp().
     */
    tearDown : function () {
        delete this.data;
    },

    //---------------------------------------------------------------------
    // Test methods - names must begin with "test"
    //---------------------------------------------------------------------

    testPop : function () {
        var Assert = Y.Assert;

        var value = this.data.pop();

        Assert.areEqual(4, this.data.length);
```

```
            Assert.areEqual(4, value);
        },

        testPush : function () {
            var Assert = Y.Assert;

            this.data.push(5);

            Assert.areEqual(6, this.data.length);
            Assert.areEqual(5, this.data[5]);
        },

        testSplice : function () {
            var Assert = Y.Assert;

            this.data.splice(2, 1, 6, 7);

            Assert.areEqual(6, this.data.length);
            Assert.areEqual(6, this.data[2]);
            Assert.areEqual(7, this.data[3]);
        }

    });

    Y.example.test.ExampleSuite = new Y.Test.Suite("Example Suite");
    Y.example.test.ExampleSuite.add(Y.example.test.DataTestCase);
    Y.example.test.ExampleSuite.add(Y.example.test.ArrayTestCase);

    //create the console
    var r = new Y.Console({
        newestOnTop : false,
        style: 'block' // to anchor in the example content
    });

    r.render('#testLogger');

    Y.Test.Runner.add(Y.example.test.ExampleSuite);

    //run the tests
    Y.Test.Runner.run();

});
```

Vector2 TestCase

```
<script type="text/javascript" src="yui-min.js"></script>
<script src="../JS/PE/class.min.js" type="text/javascript"></script>
<script src="../JS/PE/PE.js" type="text/javascript"></script>
<script src="../JS/PE/Math/Vector2.js" type="text/javascript"></script>
```

```
<body class="yui3-skin-sam  yui-skin-sam">
    <script type="text/javascript">
        YUI().use("node", "console", "test", function (Y) {

            Y.namespace("example.test");

            Y.example.test.myTestCase = new Y.Test.Case({
                name: "Vector2TestCase",
                testAddMethod: function () {
                    var Assert = Y.Assert;
                    var v1 = new PE.Vector2(10, 10);
                    var v2 = new PE.Vector2(20, 20);
                    v1.addSelf(v2);
                    Assert.areEqual(v1.x, 30);
                    Assert.areEqual(v1.y, 30);
                },
                testMultiplyScalarMethod: function () {
                    var Assert = Y.Assert;
                    var v = new PE.Vector2(10, 10);
                    var s = 5;
                    v.multiplyScalar(s);

                    Assert.areEqual(v.x, 50);
                    Assert.areEqual(v.y, 30);
                },
                testNormalizeMethod1: function () {
                    var Assert = Y.Assert;
                    var v = new PE.Vector2(100, 10);
                    var nv = v.normalize();
                    Assert.isTrue(nv.length() + 0.01 > 1);
                    Assert.isTrue(nv.length() - 0.01 < 1);
                },
```

```
                testNormalizeMethod2: function () {
                    var Assert = Y.Assert;
                    var v = new PE.Vector2(100, 10);
                    var nv = v.normalize();
                    Assert.areEqual(nv.length(), 1);
                }

            });


            Y.example.test.ExampleSuite = new Y.Test.Suite("Example
Suite");

Y.example.test.ExampleSuite.add(Y.example.test.myTestCase);
            //create the console
            var r = new Y.Console({
                width: 700,
                height: 500,
                newestOnTop: false,
                style: 'block' // to anchor in the example content
            });

            r.render();

            Y.Test.Runner.add(Y.example.test.ExampleSuite);

            //run the tests
            Y.Test.Runner.run();

        });
</script>
</body>
```

```
                testNormalizeMethod1: function () {
                    var Assert = Y.Assert;
                    var v = new PE.Vector2(100, 10);
                    var nv = v.normalize();
                    Assert.isTrue(nv.length() + 0.01 > 1);
                    Assert.isTrue(nv.length() - 0.01 < 1);
                }
```

第 4 步　集成图形化输出接口

Easeljs 平台测试

```
var stage, circle, c;
function init() {
    var canvas = document.getElementById("testCanvas");

    stage = new Stage(canvas);
    circle = new Shape();
    c = new Container();
    c.x = 100;
    c.y = 100;
    c.addChild(circle);
    stage.addChild(c);
    var g = circle.graphics;
    g.beginStroke("#444");
    g.drawCircle(0, 0, 20).lineTo(-20, 0).moveTo(0, -20).lineTo(0, 20);
    g.closePath();

    Ticker.addListener(window);
    Ticker.setFPS(60);

}
function tick() {
    c.x++;
    c.rotation += 6;
    stage.update();
}
```

```
var stage, rect, c;
function init() {
    var canvas = document.getElementById("testCanvas");

    stage = new Stage(canvas);
    rect = new Shape();
```

```
        c = new Container();
        c.x = 100;
        c.y = 100;
        c.addChild(rect);
        c.rotation = 30;
        var g = rect.graphics;
        g.beginStroke("#444");
        g.drawRect(-50, -60, 100, 120);
        g.closePath();
        stage.addChild(c);
        Ticker.addListener(window);
        Ticker.setFPS(60);

    }
    function tick() {
        c.x++;
        c.rotation += 6;
        stage.update();
    }
```

PE 集成 Easeljs

```
PE.Objects = [];
PE.ObjectRender = Class.create((function () {

    function initialize() {

    }
    function update(timeStep, stage) {
        PE.World.update(timeStep);

        var cases = {
            "Circle": drawCircle,
            "Rect": drawRect
        };

        function drawRect(obj) {
            if (obj.id === -1) {
                var o = new Container();
                o.x = obj.position.x;
```

```
                o.y = obj.position.y;
                var shape = new Shape();
                o.addChild(shape);
                o.rotation = obj.rotation;
                var g = shape.graphics;
                g.beginStroke("#444");
                g.drawRect(-obj.width / 2, -obj.height / 2, obj.width,
obj.height);
                g.closePath();
                stage.addChild(o);
                obj.id = stage.getChildIndex(o);
            }
            else {
                var o = stage.getChildAt(obj.id);
                o.x = obj.position.x;
                o.y = obj.position.y;
                o.rotation = obj.rotation;
            }
        }

        function drawCircle(obj) {
            if (obj.id === -1) {
                var r = obj.radius;
                var o = new Container();
                o.x = obj.position.x;
                o.y = obj.position.y;
                var shape = new Shape();
                o.addChild(shape);
                o.rotation = obj.rotation;
                var g = shape.graphics;
                g.beginStroke("#444");
                g.drawCircle(0, 0, r).lineTo(-r, 0).moveTo(0, -r).lineTo(0,
r);
                g.closePath();
                stage.addChild(o);
                obj.id = stage.getChildIndex(o);
            }
            else {
                var o = stage.getChildAt(obj.id);
                o.x = obj.position.x;
                o.y = obj.position.y;
                o.rotation = obj.rotation;
            }
        }
```

```
        for (var i in PE.Objects) {
            if (PE.Objects[i].type === "Circle") drawCircle(PE.Objects[i]);
            if (PE.Objects[i].type === "Rect") drawRect(PE.Objects[i]);

        }

        stage.update();
    }
    return {
        initialize: initialize,
        update: update.setStatic()

    };

} ()));
```

第 5 步　碰 撞 检 测

球与球的碰撞检测

```
        var vrn = normal.x * ((vA.x - wA * rA.y) - (vB.x - wB * rB.y)) +
normal.y * ((vA.y + wA * rA.x) - (vB.y + wB * rB.x));
```

```
        var vA = bodyA.speed;
        var vB = bodyB.speed;
        var wA = bodyA.palstance;
        var wB = bodyB.palstance;

        var r = bodyA.radius + bodyB.radius;
        var distSqr = (bodyA.position.x - bodyB.position.x) *
(bodyA.position.x - bodyB.position.x) + (bodyA.position.y -
bodyB.position.y) * (bodyA.position.y - bodyB.position.y);
        var isTouching = distSqr <= r * r ? true : false;
        var normal = new PE.Vector2();
        normal.x = bodyB.position.x - bodyA.position.x;
        normal.y = bodyB.position.y - bodyA.position.y;
        normal.normalizeSelf();
```

```
        var tangent = new PE.Vector2(normal.y, -normal.x);
        var ratio = bodyA.radius / r;
        var contactPoint = new PE.Vector2();
        contactPoint.x = bodyA.position.x + (bodyB.position.x -
bodyA.position.x) * ratio;
        contactPoint.y = bodyA.position.y + (bodyB.position.y -
bodyA.position.y) * ratio;
        var rA = PE.Vector2.sub(contactPoint, bodyA.position);
        var rB = PE.Vector2.sub(contactPoint, bodyB.position);
        var vrn = normal.x * ((vA.x - wA * rA.y) - (vB.x - wB * rB.y)) +
normal.y * ((vA.y + wA * rA.x) - (vB.y + wB * rB.x));
        if (vrn > 0&& isTouching) {
            //碰撞反应
            //省略.....
            //省略.....
            //省略.....
        }
```

球与矩形的碰撞检测

```
    function getFourVertexCoordinates() {
        var points = [];
        var v1 = new PE.Vector2(this.position.x - this.width / 2,
this.position.y - this.height / 2);
        var v2 = new PE.Vector2(this.position.x + this.width / 2,
this.position.y - this.height / 2);
        var v3 = new PE.Vector2(this.position.x + this.width / 2,
this.position.y + this.height / 2);
        var v4 = new PE.Vector2(this.position.x - this.width / 2,
this.position.y + this.height / 2);
        var rv = new PE.Vector2(this.position.x, this.position.y);
        v1.rotate(this.rotation * Math.PI / 180, rv);
        v2.rotate(this.rotation * Math.PI / 180, rv);
        v3.rotate(this.rotation * Math.PI / 180, rv);
        v4.rotate(this.rotation * Math.PI / 180, rv);
        points.push(v1);
        points.push(v2);
        points.push(v3);
        points.push(v4);
        return points;
```

```
    }
```

```
                if ((PE.Objects[i].type === "Circle" && PE.Objects[j].type
=== "Rect") || (PE.Objects[i].type === "Rect" && PE.Objects[j].type ===
"Circle")) {
                    var c, r;
                    if (PE.Objects[i].type === "Circle") {
                        c = PE.Objects[i];
                        r = PE.Objects[j];
                    }
                    else {
                        r = PE.Objects[i];
                        c = PE.Objects[j];
                    }
                    var ps = r.getFourVertexCoordinates();
                    var L1 = new PE.Line(ps[0], ps[1]);
                    var L2 = new PE.Line(ps[2], ps[3]);
                    var L3 = new PE.Line(ps[0], ps[3]);
                    var L4 = new PE.Line(ps[1], ps[2]);

                    var v1 = new PE.Vector2(ps[1].x - ps[0].x, ps[1].y -
ps[0].y);
                    var v2 = new PE.Vector2(ps[2].x - ps[1].x, ps[2].y -
ps[1].y);
                    var v3 = new PE.Vector2(ps[3].x - ps[2].x, ps[3].y -
ps[2].y);
                    var v4 = new PE.Vector2(ps[1].x - ps[3].x, ps[1].y -
ps[3].y);


                    var h1 = c.position.distanceToLine(L1);
                    var h2 = c.position.distanceToLine(L2);
                    var h3 = c.position.distanceToLine(L3);
                    var h4 = c.position.distanceToLine(L4);
                    //如果在方块里面
                    if (Math.round(h1 + h2 + h3 + h4) <=
Math.round(r.height + r.width)) {
                        if (h1 <= c.radius) rebound(v1, L1, c, r, L3, L4);
                        if (h2 <= c.radius) rebound(v2, L2, c, r, L3, L4);
                        if (h3 <= c.radius) rebound(v3, L3, c, r, L1, L2);
                        if (h4 <= c.radius) rebound(v4, L4, c, r, L1, L2);
                    }
```

```
                    else if (h1 <= r.height && h2 <= r.height) {
                        //和左右边正碰
                        if (h3 <= c.radius) rebound(v3, L3, c, r, L1, L2);
                        if (h4 <= c.radius) rebound(v4, L4, c, r, L1, L2);


                    } else if (h3 <= r.width && h4 <= r.width) {
                        //和上下两边正碰
                        if (h1 <= c.radius) rebound(v1, L1, c, r, L3, L4);
                        if (h2 <= c.radius) rebound(v2, L2, c, r, L3, L4);
                    } else {
                        //和四个角度碰撞
                        if (c.position.distanceTo(ps[0]) <= c.radius)
reboundByApex(ps[0], c, r);
                        if (c.position.distanceTo(ps[1]) <= c.radius)
reboundByApex(ps[1], c, r);

                        if (c.position.distanceTo(ps[2]) <= c.radius)
reboundByApex(ps[2], c, r);
                        if (c.position.distanceTo(ps[3]) <= c.radius)
reboundByApex(ps[3], c, r);
                    }
                }
```

```
        for (var i = 0; i < PE.Objects.length; i++) {
            for (var j = i + 1; j < PE.Objects.length; j++) {

            //....
            //....
            //....

            }
          }
```

第 6 步　方向包围盒——OBB

OBB

```
(function (window) {
    var OBB = function (centerPoint, width, height, rotation) {

        this.centerPoint = centerPoint;
```

```
        this.extents = [width / 2, height / 2];
        this.axes = [new Vector2(Math.cos(rotation), Math.sin(rotation)),
new Vector2(-1 * Math.sin(rotation), Math.cos(rotation))];

        this._width = width;
        this._height = height;
        this._rotation = rotation;
    }

    window.OBB = OBB;
})(window);
```

```
    OBB.prototype = {

        getProjectionRadius: function (axis) {
            return this.extents[0] * Math.abs(axis.dot(this.axes[0])) +
this.extents[1] * Math.abs(axis.dot(this.axes[1]));
        }

    }
```

```
(function (window) {

    var CollisionDetector = {


        detectorOBBvsOBB: function (ra, rb) {

            var nv = ra.centerPoint.sub(rb.centerPoint);

            var axisA1 = ra.axes[0];
            if (ra.getProjectionRadius(axisA1) +
rb.getProjectionRadius(axisA1) <= Math.abs(nv.dot(axisA1))) return false;

            var axisA2 = ra.axes[1];
            if (ra.getProjectionRadius(axisA2) +
rb.getProjectionRadius(axisA2) <= Math.abs(nv.dot(axisA2))) return false;

            var axisB1 = rb.axes[0];
            if (ra.getProjectionRadius(axisB1) +
rb.getProjectionRadius(axisB1) <= Math.abs(nv.dot(axisB1))) return false;
```

```
            var axisB2 = rb.axes[1];
            if (ra.getProjectionRadius(axisB2) +
rb.getProjectionRadius(axisB2) <= Math.abs(nv.dot(axisB2))) return false;

            return true;
        }

    }
    window.CollisionDetector = CollisionDetector;

})(window)
```

集成图形化接口

```
    <script src="Vector2.js" type="text/javascript"></script>
    <script src="OBB.js" type="text/javascript"></script>
    <script src="CollisionDetector.js" type="text/javascript"></script>
    <script src="../script/easel.js" type="text/javascript"></script>
    <canvas id="testCanvas" width="980" height="580">
```

```
        var OBB1, OBB1x = 100, OBB1y = 150, OBB1w = 30, OBB1h = 140,
OBB1r = 30;
        var OBB2, OBB2x = 100, OBB2y = 70, OBB2w = 40, OBB2h = 110,
OBB2r = 40;
        var OBB1, OBB2;
        var canvas;
        var stage;
        var color;

        function init() {

            canvas = document.getElementById("testCanvas");
            stage = new Stage(canvas);

            Ticker.addListener(window);
        }
```

```
        function tick() {
            stage.removeAllChildren();

            OBB1r += 2;
            OBB2r += 1;
            OBB1 = new OBB(new Vector2(OBB1x, OBB1y), OBB1w, OBB1h,
OBB1r * Math.PI / 180);
            OBB2 = new OBB(new Vector2(OBB2x, OBB2y), OBB2w, OBB2h,
OBB2r * Math.PI / 180);
            var r = CollisionDetector.detectorOBBvsOBB(OBB1, OBB2);
            if (r) {
                color = "red"
            }
            else {
                color = "#00F";
            }
            OBB1 = new Container();
            stage.addChild(OBB1);
            OBB1.x = OBB1x;
            OBB1.y = OBB1y;
            var frame1 = new Shape();
            frame1.graphics.beginFill(color).drawRect(0, 0, OBB1w,
OBB1h);
            frame1.rotation = OBB1r;
            frame1.regX = OBB1w / 2;
            frame1.regY = OBB1h / 2;
            OBB1.addChild(frame1);

            OBB2 = new Container();
            stage.addChild(OBB2);
            OBB2.x = OBB2x;
            OBB2.y = OBB2y;
            var frame2 = new Shape();
            frame2.graphics.beginFill(color).drawRect(0, 0, OBB2w,
OBB2h);
            frame2.rotation = OBB2r;
            frame2.regX = OBB2w / 2;
            frame2.regY = OBB2h / 2;
            OBB2.addChild(frame2);

            stage.update();
        }
        init();
```

第 7 步　碰 撞 反 应

```
    function collisionCCHandling(bodyA, bodyB) {
        var vA = bodyA.speed;
        var vB = bodyB.speed;
        var wA = bodyA.palstance;
        var wB = bodyB.palstance;


        var r = bodyA.radius + bodyB.radius;
        var distSqr = (bodyA.position.x - bodyB.position.x) *
(bodyA.position.x - bodyB.position.x) + (bodyA.position.y -
bodyB.position.y) * (bodyA.position.y - bodyB.position.y);
        var isTouching = distSqr <= r * r ? true : false;
        var normal = new PE.Vector2();
        normal.x = bodyB.position.x - bodyA.position.x;
        normal.y = bodyB.position.y - bodyA.position.y;
        normal.normalizeSelf();
        var tangent = new PE.Vector2(normal.y, -normal.x);
        var ratio = bodyA.radius / r;
        var contactPoint = new PE.Vector2();
        contactPoint.x = bodyA.position.x + (bodyB.position.x -
bodyA.position.x) * ratio;
        contactPoint.y = bodyA.position.y + (bodyB.position.y -
bodyA.position.y) * ratio;

        var rA = PE.Vector2.sub(contactPoint, bodyA.position);
        var rB = PE.Vector2.sub(contactPoint, bodyB.position);
        var vrn = normal.x * ((vA.x - wA * rA.y) - (vB.x - wB * rB.y)) +
normal.y * ((vA.y + wA * rA.x) - (vB.y + wB * rB.x));
        var vrt = tangent.x * ((vA.x - wA * rA.y) - (vB.x - wB * rB.y)) +
tangent.y * ((vA.y + wA * rA.x) - (vB.y + wB * rB.x));
        if (isTouching && vrn > 0) {
            var normalMass = 1 / (bodyA.getInvMass() +
bodyB.getInvMass());
            var restitution = (bodyA.restitution + bodyB.restitution) / 2;
            var normalImpulse = -normalMass * vrn * (1 + restitution);
            bodyA.speed.x += normalImpulse * normal.x *
bodyA.getInvMass();
            bodyA.speed.y += normalImpulse * normal.y *
```

```
bodyA.getInvMass();
            bodyB.speed.x -= normalImpulse * normal.x *
bodyB.getInvMass();
            bodyB.speed.y -= normalImpulse * normal.y *
bodyB.getInvMass();

            if (vrt != 0) {

                var tangentImpulse = normalImpulse * (bodyA.friction +
bodyB.friction) / 2;

                if (vrt < 0)
                    tangentImpulse *= -1;

                var Ix = tangentImpulse * tangent.x;
                var Iy = tangentImpulse * tangent.y;
                bodyA.speed.x += Ix * bodyA.getInvMass();
                bodyA.speed.y += Iy * bodyA.getInvMass();

                wA += (rA.x * Iy - rA.y * Ix) *
bodyA.getInvRotationalInertia();

                bodyB.speed.x -= Ix * bodyB.getInvMass();
                bodyB.speed.y -= Iy * bodyB.getInvMass();
                wB -= (rB.x * Iy - rB.y * Ix) *
bodyB.getInvRotationalInertia();

                bodyA.palstance = wA;
                bodyB.palstance = wB;
            }
        }

    }
```

```
        var contactPoint = line.getVerticalCrossoverPoint(bodyA.position);
        var d1 = contactPoint.distanceToLine(L1);
        var d2 = contactPoint.distanceToLine(L2);
        var d;
        if (d1 > d2) { d = d2; } else { d = d1 };


        var v = PE.Vector2.sub(contactPoint, bodyA.position);
```

```
        v.multiplyScalar(L1.length / (bodyA.radius * 2));
        var tempCenter = PE.Vector2.add(contactPoint, v);

        var vA = bodyA.speed;
        var vB = bodyB.speed;
        var wA = bodyA.palstance;
        var wB = bodyB.palstance;

        var normal = new PE.Vector2();
        normal.x = tempCenter.x - bodyA.position.x;
        normal.y = tempCenter.y - bodyA.position.y;
        normal.normalizeSelf();
        var tangent = new PE.Vector2(normal.y, -normal.x);
        var rA = PE.Vector2.sub(contactPoint, bodyA.position);
        var rB = PE.Vector2.sub(contactPoint, bodyB.position);
        var vrn = normal.x * ((vA.x - wA * rA.y) - (vB.x - wB * rB.y)) +
normal.y * ((vA.y + wA * rA.x) - (vB.y + wB * rB.x));
        var vrt = tangent.x * ((vA.x - wA * rA.y) - (vB.x - wB * rB.y)) +
tangent.y * ((vA.y + wA * rA.x) - (vB.y + wB * rB.x));
        if (vrn > 0) {

            var normalMass = 1 / (bodyA.getInvMass() +
bodyB.getInvMass() / (d * 2 / (d1 + d2)));
            var restitution = (bodyA.restitution + bodyB.restitution) / 2;
            var normalImpulse = -normalMass * vrn * (1 + restitution);
            bodyA.speed.x += normalImpulse * normal.x *
bodyA.getInvMass();
            bodyA.speed.y += normalImpulse * normal.y *
bodyA.getInvMass();
            bodyB.speed.x -= normalImpulse * normal.x *
bodyB.getInvMass();
            bodyB.speed.y -= normalImpulse * normal.y *
bodyB.getInvMass();

            wB -= (rB.x * normalImpulse * normal.y - rB.y *
normalImpulse * normal.x) * bodyB.getInvRotationalInertia();
            bodyB.palstance = wB;


            if (vrt != 0) {

                var tangentImpulse = normalImpulse * (bodyA.friction +
bodyB.friction) / 2;
```

```
                if (vrt < 0)
                    tangentImpulse *= -1;

                var Ix = tangentImpulse * tangent.x;
                var Iy = tangentImpulse * tangent.y;

                bodyA.speed.x += Ix * bodyA.getInvMass();
                bodyA.speed.y += Iy * bodyA.getInvMass();

                wA += (rA.x * Iy - rA.y * Ix) *
bodyA.getInvRotationalInertia();

                bodyB.speed.x -= Ix * bodyB.getInvMass();
                bodyB.speed.y -= Iy * bodyB.getInvMass();

                wB -= (rB.x * Iy - rB.y * Ix) *
bodyB.getInvRotationalInertia();

                bodyA.palstance = wA;
                bodyB.palstance = wB;
            }
        }
```

第 8 步　重 叠 处 理

圆形与圆形的重叠处理

```
            var distance = bodyA.position.distanceTo(bodyB.position);
            var cosAngle = (bodyA.position.x - bodyB.position.x) /
distance;
            var sinAngle = (bodyA.position.y - bodyB.position.y) / distance;

            var tempX = (1 + bodyA.radius + bodyB.radius - distance) *
cosAngle / 2;
            var tempY = (1 + bodyA.radius + bodyB.radius - distance) *
sinAngle / 2;
            bodyA.position.x += tempX;
            bodyA.position.y += tempY;
```

```
            bodyB.position.x -= tempX;
            bodyB.position.y -= tempY;
```

圆形与矩形的重叠处理

```
    function resolveOverlap(c, r, p) {

            c.position.x += (c.radius - c.position.distanceTo(p)) *
(c.position.x - p.x) / c.position.distanceTo(p);
            c.position.y += (c.radius - c.position.distanceTo(p)) *
(c.position.y - p.y) / c.position.distanceTo(p);

    }
```

第 9 步　贴　图

重构 Body 类

```
    function initialize() {
        this.id = -1;
        this.bodyType = PE.BodyType.Dynamic;
        this.ignoreGravity = true;
        this.enabled = true;
        this.friction = 0.2;
        this.restitution = 1.0;

        this.position = PE.Vector2.Zero;
        this.mass = 1;
        this.speed = PE.Vector2.Zero;
        this.force = PE.Vector2.Zero;
        this.rotation = 0;
        this.palstance = 0;
        this.imgUrl = "";
        PE.World.addBody(this);
```

```
    }
```

```
        function drawCircle(obj) {
            if (obj.id === -1) {
                var r = obj.radius;
                var o = new Container();
                o.x = obj.position.x;
                o.y = obj.position.y;
                var shape = new Shape();
                o.addChild(shape);
                o.rotation = obj.rotation;
                var g = shape.graphics;
                g.beginStroke("#444");
                g.drawCircle(0, 0, r).lineTo(-r, 0).moveTo(0, -r).lineTo(0,
r);
                g.closePath();
                stage.addChild(o);
                obj.id = stage.getChildIndex(o);
            }
            else {
                var o = stage.getChildAt(obj.id);
                o.x = obj.position.x;
                o.y = obj.position.y;
                o.rotation = obj.rotation;
            }
        }
```

```
        function drawCircle(obj) {
            if (obj.id === -1) {
                var r = obj.radius;
                var o = new Container();
                o.x = obj.position.x;
                o.y = obj.position.y;
                o.rotation = obj.rotation;
                if (obj.imgUrl === "") {
                    var shape = new Shape();
                    o.addChild(shape);
                    var g = shape.graphics;
                    g.beginStroke("#444");
                    g.drawCircle(0, 0, r).lineTo(-r, 0).moveTo(0,
-r).lineTo(0, r);
                    g.closePath();
                }
```

```
            else {
                var img = new Bitmap(obj.imgUrl);
                img.x = -obj.radius;
                img.y = -obj.radius;
                o.addChild(img);
            }
            stage.addChild(o);
            obj.id = stage.getChildIndex(o);
        }
        else {
            var o = stage.getChildAt(obj.id);
            o.x = obj.position.x;
            o.y = obj.position.y;
            o.rotation = obj.rotation;
        }
    }
```

```
                var img = new Bitmap(obj.imgUrl);
                img.x = -obj.radius;
                img.y = -obj.radius;
```

物理引擎作品展示一

```
<script src="JS/PE/sylvester.js" type="text/javascript"></script>
<script src="JS/PE/easel.js" type="text/javascript"></script>
<script src="JS/PE/class.min.js" type="text/javascript"></script>
<script src="JS/PE/PE.js" type="text/javascript"></script>
<script src="JS/PE/ObjectRender.js" type="text/javascript"></script>
<script src="JS/PE/Shape/Body.js" type="text/javascript"></script>
<script src="JS/PE/Shape/Circle.js" type="text/javascript"></script>
<script src="JS/PE/Math/Vector2.js" type="text/javascript"></script>
<script src="JS/PE/Shape/Rect.js" type="text/javascript"></script>
<script src="JS/PE/CM2.js" type="text/javascript"></script>
<script src="JS/PE/Math/Line.js" type="text/javascript"></script>
<script src="JS/PE/Math/Math.js" type="text/javascript"></script>
<script src="JS/PE/World.js" type="text/javascript"></script>
<script src="JS/PE/BodyCounter.js" type="text/javascript"></script>
```

```
    var stage, circle, rect, m = PE.Math;
    function init() {
        var canvas = document.getElementById("testCanvas");
        stage = new Stage(canvas);
        PE.World.gravity = new PE.Vector2(0, 400);
        for (var i = 0; i < 40; i++) {
            circle = new PE.Circle();
            circle.speed = new PE.Vector2(m.getRandomNumber(100,
200), m.getRandomNumber(100, 200));
            circle.radius = 20;
            circle.mass = 20;
            circle.friction = 0.4;
            circle.restitution = 0.2;
            circle.palstance = 0;
            circle.position = new PE.Vector2(m.getRandomNumber(1, 600),
m.getRandomNumber(4, 600));
        }

        rect = new PE.Rect();
        rect.position = new PE.Vector2(240, 200);
        rect.width = 240;
        rect.height = 240;
        rect.bodyType = PE.BodyType.Static;
        rect.palstance = Math.PI / 6;
        Ticker.addListener(window);
        Ticker.setFPS(60);
    }
```

```
    function tick() {
        PE.ObjectRender.update(1.0 / 60, stage);
    }
```

物理引擎作品展示二

```
    var stage, circle, rect, m = PE.Math;
    function init() {
        var canvas = document.getElementById("testCanvas");
```

```
        stage = new Stage(canvas);
        PE.World.gravity = new PE.Vector2(0, 0);

        circle = new PE.Circle();
        circle.radius = 20;
        circle.mass = 20;
        circle.friction = 0.1;
        circle.restitution = 1;
        circle.position = new PE.Vector2(290, 110);
        circle.speed = new PE.Vector2(-140, 0);


        rect = new PE.Rect();
        rect.rotation = 4;
        rect.friction = 0.1;
        rect.position = new PE.Vector2(140, 70);
        rect.width = 40;
        rect.speed = new PE.Vector2(0, 0);
        rect.height = 140;
        rect.restitution = 1;
        rect.mass = 190;
        Ticker.addListener(window);
        Ticker.setFPS(60);
    }
    function tick() {
        PE.ObjectRender.update(1.0 / 60, stage);
    }
```

物理引擎作品展示三

```
        var fpsLabel;
        var stage;
        var m = PE.Math;
        function init() {
            var canvas = document.getElementById("testCanvas");

            stage = new Stage(canvas);
```

```
            PE.World.gravity = new PE.Vector2(0, 400);

            fpsLabel = new Text("-- fps", "bold 14px Arial", "#000");
            stage.addChild(fpsLabel);
            fpsLabel.x = 10;
            fpsLabel.y = 20;
            for (var i = 0; i < 11; i++) {
                var circle = new PE.Circle();
                circle.speed = new PE.Vector2(m.getRandomNumber(0,
200), m.getRandomNumber(0, 200));
                circle.radius = 15;
                circle.mass = 20;
                circle.palstance = 0;
                circle.friction = 0.5;
                circle.restitution = 0.4;
                circle.position = new
PE.Vector2(m.getRandomNumber(390, 400), m.getRandomNumber(0,
20));
            }

            var rect1 = new PE.Rect();
            rect1.position = new PE.Vector2(240, 320);
            rect1.width = 440;
            rect1.height = 20;
            rect1.bodyType = PE.BodyType.Static;
            rect1.friction = 0.1;
            rect1.palstance = 0;
            rect1.restitution = 0.4;
            rect1.rotation = 16;

            var rect2 = new PE.Rect();
            rect2.position = new PE.Vector2(490, 170);
            rect2.width = 440;
            rect2.height = 20;
            rect2.restitution = 0.4;
            rect2.bodyType = PE.BodyType.Static;
            rect2.friction = 0.1;
            rect2.palstance = 0;
            rect2.rotation = -16;

            var rect3 = new PE.Rect();
            rect3.position = new PE.Vector2(560, 440);
            rect3.width = 440;
            rect3.height = 20;
```

```
            rect3.bodyType = PE.BodyType.Static;
            rect3.friction = 0.1;
            rect3.palstance = 0;
            rect3.rotation = -16;
            rect3.restitution = 0.4;

            Ticker.addListener(window);
            Ticker.setFPS(60);
        }
        function tick() {
            PE.ObjectRender.update(1.0 / 60, stage);

            fpsLabel.text = Math.round(Ticker.getMeasuredFPS()) + " fps";
        }
```

```
            for (var i = 0; i < 11; i++) {
                var circle = new PE.Circle();
                circle.speed = new PE.Vector2(m.getRandomNumber(0,
200), m.getRandomNumber(0, 200));
                circle.radius = 45;
                circle.mass = 20;
                circle.imgUrl = "img/ball.png";
                circle.palstance = 0;
                circle.friction = 0.5;
                circle.restitution = 0.4;
                circle.position = new
PE.Vector2(m.getRandomNumber(390, 400), m.getRandomNumber(0,
20));
            }
```

第 6 章　游戏开发全程实录

6.2　框 架 搭 建

简介

```
<script src="../lib/easel.js"></script>
```

创建舞台

```
<canvas id="gameCanvas" width="980" height="580"></canvas>
```

```
var canvas = document.createElement("canvas");
canvas.width = 480;
canvas.height = 820;
```

```
    var canvas = document.getElementById("gameCanvas");
    var stage = new Stage(canvas);
```

游戏循环

```
    function startGame() {
        //start game timer
        Ticker.addListener(window);
    }
    function tick() {
        //.......
        //code here
```

```
        //.....
        stage.update();
    }
```

hello world

```
    var stage,text;
    function init() {

        var canvas = document.getElementById("gameCanvas");
        //创建舞台
        stage = new Stage(canvas);

        // 创建一个text对象
        text = new Text("Hello World!", "36px Arial", "#FFF");

        //把text对象添加到舞台当中
        stage.addChild(text);

        //设置text在canvas中的位置
        text.x = 100;
        text.y = 100;

        Ticker.addListener(window);
    }

    function tick() {
        text.x++;
        stage.update();
    }
```

```
    var stage,bmp;
    function init() {

        var canvas = document.getElementById("gameCanvas");
        //创建舞台
        stage = new Stage(canvas);

        //创建一个位图对象
        bmp = new Bitmap("img/test.png");
```

```
        stage.addChild(bmp);

        //把位图对象添加到舞台当中
        stage.addChild(bmp);

        Ticker.addListener(window);
    }

    function tick() {
        //位图对象的x坐标不断增加（右移）
        bmp.x++;
        stage.update();
    }
```

6.3 资 源 加 载

PxLoader 的使用

```
<script type="text/javascript" src="js/PxLoader.js"></script>
<script type="text/javascript" src="js/PxLoaderImage.js"></script>
<script type="text/javascript" src="js/PxLoaderSound.js"></script>
```

```
    //把三张图片放入下载队列当中，当执行 loader.start()的时候，图片开始下载
    var loader = new PxLoader(),
      backgroundImg = loader.addImage('images/headerbg.jpg'),
      treesImg = loader.addImage('images/trees.png'),
      ufoImg = loader.addImage('images/ufo.png');

    //加载完成立刻执行回调
    loader.addCompletionListener(function () {
        var canvas = document.getElementById('sample1-canvas'),
        ctx = canvas.getContext('2d');
        ctx.drawImage(backgroundImg, 0, 0);
        ctx.drawImage(treesImg, 0, 104);
        ctx.drawImage(ufoImg, 360, 50);
    });
```

```
    //开始下载图片
    loader.start();
```

Easeljs 集成 PxLoader

```
    function loadImage() {

        var loader = new PxLoader();
        backgroundImg = loader.addImage('backdrop.png');
        plasmaImg = loader.addImage('plasma.png');
        enemyImage = loader.addImage('mine.png');
        explodeImage = loader.addImage('explosion2-64.png');
        shipBmp = loader.addImage('ship.png');
        gridImage = loader.addImage('grid.png');

        loader.addCompletionListener(function () {
            init();
        });


        //每下载完一张图片，执行下面方法，用于显示下载进度
        loader.addProgressListener(function (e) {
            // console.log(e.resource.imageNumber +"___"+
e.completedCount + ' / ' + e.totalCount);

        });

        loader.start();
    }


    function init() {
        canvas = document.getElementById("canvas");
        stage = new Stage(canvas);

        bgBmp = new Bitmap(backgroundImg);
        plasmaBmp = new Bitmap(plasmaImg);
        enemyBmp = new Bitmap(enemyImage);
        girdBmp = new Bitmap(gridImage);
        stage.addChild(bgBmp);
```

```
        stage.addChild(plasmaBmp);
        stage.addChild(enemyBmp);
        stage.addChild(girdBmp);
        //.....
        //.....
        //.....
    }
```

Easeljs 集成 loading

```
function main() {
    canvas = document.getElementById("canvas");
    stage = new Stage(canvas);
    loadingTxt = new Bitmap(loadingCanvas);
    loadingTxt.y = 200;
    stage.addChild(loadingTxt);
    Ticker.setFPS(60);
    Ticker.addListener(window);
    loadImage ();
}

function tick() {

    stage.update();
}
```

6.4 菜 单 制 作

技术细节

类型一：精确到像素

```
        container = new Container();
        container.x = 133;
        container.y = 260;
        container.addChild(txt);
```

```
        txt = new Text("PLAY", "bold 36px Arial", "#FFF");
        txt.textBaseline = "top";
        container.addChild(txt);
        container.onClick = function (evt) {
            beginGame();
        }
        container.onMouseOver = function (e) {
            txt.color = "#FF7E00";
            stage.update();
        }
        container.onMouseOut = function (e) {
            txt.color = "#fff";
            stage.update();
        }
        stage.addChild(container);
```

类型二：精确到矩形区域

```
        container = new Container();
        container.x = 133;
        container.y = 260;
        target = new Shape();
        target.alpha = 0.01;
        target.graphics.beginFill("#3A5C68").drawRect(-10, -10, 105,
50).beginFill("#FFF");
        container.addChild(target);
        txt = new Text("PLAY", "bold 36px Arial", "#FFF");
        txt.textBaseline = "top";
        container.addChild(txt);
        container.onClick = function (evt) {
            beginGame();
        }
        container.onMouseOver = function (e) {
            txt.color = "#FF7E00";
            stage.update();
        }
        container.onMouseOut = function (e) {
            txt.color = "#fff";
            stage.update();
        }
        stage.addChild(container);
```

完善菜单

```
        var aboutContainer = new Container();
        aboutContainer.x = 113;
        aboutContainer.y = 310;
        var aboutTarget = new Shape();
        aboutTarget.alpha = 0.01;
        aboutTarget.graphics.beginFill("#3A5C68").drawRect(0, 0, 131,
40);
        aboutContainer.addChild(aboutTarget);
        var aboutTxt = new Text("ABOUT", "bold 36px Arial", "#FFF");
        aboutTxt.textBaseline = "top";
        aboutContainer.addChild(aboutTxt);
        aboutContainer.onClick = function (evt) {
            //....
            //....
            //....
        }
        aboutContainer.onMouseOver = function (e) {
            aboutTxt.color = "#FF7E00";
            stage.update();
        }
        aboutContainer.onMouseOut = function (e) {
            aboutTxt.color = "#fff";
            stage.update();
        }
        stage.addChild(aboutContainer);

        container = new Container();
        container.x = 133;
        container.y = 360;
        target = new Shape();
        target.alpha = 0.01;
        target.graphics.beginFill("#3A5C68").drawRect(-10, -10, 105, 50);
        container.addChild(target);
        txt = new Text("EXIT", "bold 36px Arial", "#FFF");
        txt.textBaseline = "top";
        container.addChild(txt);
        container.onClick = function (evt) {
            //....
            //....
            //....
```

```
        }
        container.onMouseOver = function (e) {
            txt.color = "#FF7E00";
            stage.update();
        }
        container.onMouseOut = function (e) {
            txt.color = "#fff";
            stage.update();
        }
        stage.addChild(container);

        //...
        //...
        //省略
        //...
        //...
```

6.5 对 象 建 立

菜单项

```
        var aboutContainer = new Container();
        aboutContainer.x = 113;
        aboutContainer.y = 310;
        var aboutTarget = new Shape();
        aboutTarget.alpha = 0.01;
        aboutTarget.graphics.beginFill("#3A5C68").drawRect(0, 0, 131,
40).beginFill("#FFF");
        aboutContainer.addChild(aboutTarget);
        var aboutTxt = new Text("ABOUT", "bold 36px Arial", "#FFF");
        aboutTxt.textBaseline = "top";
        aboutContainer.addChild(aboutTxt);
        aboutContainer.onClick = function (evt) {
            alert("under construction. ： ） ");
        }
        aboutContainer.onMouseOver = function (e) {
            aboutTxt.color = "#FF7E00";
            stage.update();
```

```
        }
        aboutContainer.onMouseOut = function (e) {
            aboutTxt.color = "#fff";
            stage.update();
        }
        stage.addChild(aboutContainer);
```

```
        var menuItem = new MenuItem(new Vector2(113, 310), "ABOUT",
"bold 36px Arial", "#FFF");
        menuItem.onClick = function (evt) {
            alert("under construction. ：）");
        }
        menuItem.onMouseOver = function (e) {
            aboutTxt.color = "#FF7E00";
            stage.update();
        }
        menuItem.onMouseOut = function (e) {
            aboutTxt.color = "#fff";
            stage.update();
        }
        stage.addChild(aboutContainer);
```

```
(function (window) {

    var MenuItem = function (x, y, targetColor, targetWidth, targetHeight,
text, font, color) {

        this.initialize(x, y, targetColor, targetWidth, targetHeight, text, font,
color);
    }

    MenuItem.prototype = new Container();

    MenuItem.prototype.Container_initialize =
MenuItem.prototype.initialize;

    MenuItem.prototype.initialize = function (x, y, targetColor, targetWidth,
targetHeight, text, font, color) {

        this.Container_initialize();
        //设置容器的x、y坐标
        this.x = x;
        this.y = y;
```

```
        //创建目标区域对象
        var target = new Shape();
        //设置目标区域透明度
        target.alpha = 0.01;
        //绘制目标区域范围
        target.graphics.beginFill(targetColor).drawRect(0, 0, targetWidth,
targetHeight);
        //添加目标区域到的容器
        this.addChild(target);
        //创建文字对象
        this.txt = new Text(text, font, color);
        this.txt.textBaseline = "top";
        //添加文字到容器
        this.addChild(this.txt);

    }

    //设置容器中文字的字体颜色
    MenuItem.prototype.setTxtColor = function (color) {

        this.txt.color = color;

    }
```

```
        var aboutMenu = new MenuItem(113, 320, "#3A5C68", 128, 38,
"ABOUT", "bold 36px Arial", "#FFF");
        aboutMenu.onClick = function (evt) {
            alert("under construction. ：）");
            SoundJS.play("begin", SoundJS.INTERUPT_LATE);

        }
        aboutMenu.onMouseOver = function (e) {
            this.setTxtColor("#FF7E00");
            SoundJS.play("begin", SoundJS.INTERUPT_LATE);
            stage.update();
        }
        aboutMenu.onMouseOut = function (e) {
            this.setTxtColor("#FFF");
            stage.update();
        }
        stage.addChild(aboutMenu);
```

```
        var playMenu = new MenuItem(133, 260, "#3A5C68", 95, 40,
"PLAY", "bold 36px Arial", "#FFF");
        playMenu.onClick = function (evt) {
            gameStartTag = true;
            this.visible = false;
            stage.removeChild(aboutMenu);
            generateMines(10);
            SoundJS.play("begin", SoundJS.INTERUPT_LATE);
        }
        playMenu.onMouseOver = function (e) {
            this.setTxtColor("#FF7E00");
            SoundJS.play("begin", SoundJS.INTERUPT_LATE);
            stage.update();
        }
        playMenu.onMouseOut = function (e) {
            this.setTxtColor("#FFF");
            stage.update();
        }
        stage.addChild(playMenu);
```

飞船

```
(function (window) {

    function Ship(spriteSheetImage) {
        this.initialize(spriteSheetImage);
    }
    Ship.prototype = new BitmapAnimation();

    Ship.prototype.Container_initialize = Ship.prototype.initialize;

    Ship.prototype.initialize = function (spriteSheetImage) {
        var spriteSheet = new SpriteSheet({
            images: [spriteSheetImage],
            frames: { width: 24, height: 24, regX: 12, regY: 12 },
            animations: {
                fire: [0, 3, "fire"]
            }
        });
        this.Container_initialize(spriteSheet);
        this.gotoAndPlay("fire");
```

```
        this.onAnimationEnd = function () {
            this.gotoAndStop("fire");
        }
        this.direction = 0;
        this.x = 180;
        this.y = 560;


    }
    Ship.prototype.fire = function () {
        this.gotoAndPlay("fire");
    }


    Ship.prototype.tick = function () {
        this.x += this.vX;
        this.y += this.vY;
    }

    window.Ship = Ship;
} (window));
```

```
        var ship = new Ship(shipImage);
        stage.addChild(ship);
```

敌机

```
    var Enemy = function (v, image) {
        this.initialize(v, image);
    }
```

```
    Enemy.prototype = new Container();
    Enemy.prototype.Container_initialize = Enemy.prototype.initialize;
    Enemy.prototype.initialize = function (v, image) {

        this.Container_initialize();
        var bmp = new Bitmap(image);
        this.addChild(bmp);
        this.velocity = v;
    }
```

```
    Enemy.prototype.tick = function () {
        this.x += this.velocity.x;
        this.y += this.velocity.y;
    }
```

子弹

```
(function (window) {
    var Plasma = function (velocity, image) {

        this.initialize(velocity, image);
    }

    Plasma.prototype = new Container();

    Plasma.prototype.Container_initialize = Plasma.prototype.initialize;
    Plasma.prototype.initialize = function (velocity, image) {

        this.Container_initialize();
        var bmp = new Bitmap(image);
        bmp.regX = bmp.image.width/2;
        bmp.regY = bmp.image.height;

        this.addChild(bmp);
        this.velocity = velocity;
    }

    Plasma.prototype.tick = function () {

        this.x += this.velocity.x;
        this.y += this.velocity.y;

    }

    window.Plasma = Plasma;
} (window));
```

```
        bmp.regX = bmp.image.width/2;
        bmp.regY = bmp.image.height;
```

6.6 碰 撞 检 测

碰撞规则

```
        bmp.regX = bmp.image.width/2;
        bmp.regY = bmp.image.height;
```

```
var pp = new Vector2(plasmas[i].x + 48 * Math.sin(plasmas[i].rotation *
Math.PI / 180), plasmas[i].y - 48 * Math.cos(plasmas[i].rotation * Math.PI
/ 180));
```

```
var ep = new Vector2(enemys[j].x + 16, enemys[j].y + 16);
```

```
if (pp.distanceTo(ep) < 8) {

}
```

```
                //处理碰撞检测
                for (var i = 0; i < plasmas.length; i++) {
                    for (var j = 0; j < enemys.length; j++) {
                            var ep = new Vector2(enemys[j].x + 16,
enemys[j].y + 16);
                            var pp = new Vector2(plasmas[i].x + 48 *
Math.sin(plasmas[i].rotation * Math.PI / 180), plasmas[i].y - 48 *
Math.cos(plasmas[i].rotation * Math.PI / 180));

                            if (pp.distanceTo(ep) < 8) {
                              //碰撞处理
                            }
                        }
                    }
                }
```

爆炸

```
        var spriteSheet = new SpriteSheet({
            images: [explodeImage],
            frames: { width: 64, height: 64, regX: 32, regY: 32 },
            animations: {
                explode: [0, 6, "explode"]
            }
        });
        explode = new BitmapAnimation(spriteSheet);
        explode.onAnimationEnd = function () {
            stage.removeChild(this);
        }
        stage.addChild(explode);
```

6.7 游 戏 音 效

加载音频文件

```
    function loadSoundThenImage() {

        var filetype;
        agent = navigator.userAgent.toLowerCase();

        if (agent.indexOf("chrome") > -1) {
            filetype = ".mp3";
        } else if (agent.indexOf("opera") > -1) {
            filetype = ".ogg";
        } else if (agent.indexOf("firefox") > -1) {
            filetype = ".ogg";
        } else if (agent.indexOf("safari") > -1) {
            filetype = ".mp3";
        } else if (agent.indexOf("msie") > -1) {
            filetype = ".mp3";
        }
```

```
            SoundJS.onLoadQueueComplete = loadImage;
            SoundJS.addBatch([
        { name: "begin", src: "mp3/assets/Game-Spawn" + filetype,
instances: 1 },
        { name: "break", src: "mp3/assets/Game-Break" + filetype,
instances: 6 },
        { name: "death", src: "mp3/assets/Game-Death" + filetype,
instances: 1 },
        { name: "laser", src: "mp3/assets/Game-Shot" + filetype, instances:
6 },
        { name: "music", src:
"mp3/assets/18-machinae_supremacy-lord_krutors_dominion" + filetype,
instances: 1}]);

    }
```

```
        aboutMenu.onMouseOver = function (e) {
            this.setTxtColor("#FF7E00");
            SoundJS.play("begin", SoundJS.INTERUPT_LATE);
            stage.update();
        }
        aboutMenu.onMouseOut = function (e) {
            this.setTxtColor("#FFF");
            stage.update();
        }
```

```
        playMenu.onClick = function (evt) {
            gameStartTag = true;
            this.visible = false;
            stage.removeChild(aboutMenu);
            generateMines(10);
            SoundJS.play("begin", SoundJS.INTERUPT_LATE);
        }
```

```
                if (shootHeld === true) {
                    var v = new Vector2(-Math.sin(ship.rotation * (Math.PI
/ -180)) * 10, -Math.cos(ship.rotation * (Math.PI / -180)) * 10);
                    var p = new Plasma(v,   plasmaImg);
                    p.x = ship.x;
```

```
                    p.y = ship.y;
                    p.rotation = ship.rotation;
                    plasmas.push(p);
                    stage.addChild(p);
                    SoundJS.play("laser", SoundJS.INTERUPT_LATE);
                    ship.fire();
                }
```

```
    for (var i = 0; i < plasmas.length; i++) {
                    for (var j = 0; j < enemys.length; j++) {

                        if (plasmas[i]) {
                            var ep = new Vector2(enemys[j].x + 16,
enemys[j].y + 16);
                            var pp = new Vector2(plasmas[i].x + 48 *
Math.sin(plasmas[i].rotation * Math.PI / 180), plasmas[i].y - 48 *
Math.cos(plasmas[i].rotation * Math.PI / 180));

                            if (pp.distanceTo(ep) < 8) {
                                score += MathHelp.getRandomNumber(10,
50);

                                explode.x = ep.x;
                                explode.y = ep.y;
                                stage.removeChild(enemys[j]);
                                stage.removeChild(plasmas[i]);
                                plasmas.splice(i--, 1);
                                enemys.splice(j--, 1);

                                var explodeClone = explode.clone();
                                explodeClone.gotoAndPlay("explode");
                                explodeClone.onAnimationEnd = function ()
{
                                    stage.removeChild(this);
                                };
                                stage.addChild(explodeClone);
                                SoundJS.play("break",
SoundJS.INTERUPT_LATE);
                            }
                        }
                    }
                }
```

```
        SoundJS.play("music", null, 0.2, true);
```

6.8 键 盘 控 制

键盘控制

```
    var shootHeld;
    var lfHeld;
    var rtHeld;
document.onkeydown = handleKeyDown;
document.onkeyup = handleKeyUp;
var KEYCODE_LEFT = 37;
var KEYCODE_RIGHT = 39;
var KEYCODE_A = 65;
var KEYCODE_D = 68;
var KEYCODE_J = 74;

function handleKeyDown(e) {
    if (!e) { var e = window.event; }

    switch (e.keyCode) {
        case KEYCODE_J: shootHeld = true; break;
        case KEYCODE_A: lfHeld = true; break;
        case KEYCODE_LEFT: lfHeld = true; break;
        case KEYCODE_D: rtHeld = true; break;
        case KEYCODE_RIGHT: rtHeld = true; break;
    }
}

function handleKeyUp(e) {
    if (!e) { var e = window.event; }

    switch (e.keyCode) {
        case KEYCODE_J: shootHeld = false; break;
        case KEYCODE_A: lfHeld = false; break;
        case KEYCODE_LEFT: lfHeld = false; break;
        case KEYCODE_D: rtHeld = false; break;
        case KEYCODE_RIGHT: rtHeld = false; break;
```

```
        }
    }
```

```
    function tick() {
        if (gameStartTag) {

            //生成敌机
            if (enemys.length === 0) {
                mineCount += 10;
                generateMines(mineCount);
            }



            //从舞台中移除远去敌机
            for (var j = 0; j < enemys.length; j++) {
                if (enemys[j].x > canvas.width + 100 || enemys[j].x <
-100 || enemys[j].y > canvas.height + 20 || enemys[j].y < -100) {
                    stage.removeChild(enemys[j]);
                    enemys.splice(j--, 1);
                }
            }

            //从舞台中移除远去的子弹
            for (var i = 0; i < plasmas.length; i++) {
                var pp = plasmas[i];
                if (pp.x > canvas.width + 100 || pp.x < -100 || pp.y >
canvas.height + 100 || pp.y < -100) {
                    stage.removeChild(plasmas[i]);
                    plasmas.splice(i--, 1);

                }
            }

            //处理碰撞检测
            for (var i = 0; i < plasmas.length; i++) {
                for (var j = 0; j < enemys.length; j++) {

                    if (plasmas[i]) {
                        var ep = new Vector2(enemys[j].x + 16,
enemys[j].y + 16);
                        var pp = new Vector2(plasmas[i].x + 48 *
Math.sin(plasmas[i].rotation * Math.PI / 180), plasmas[i].y - 48 *
```

```
Math.cos(plasmas[i].rotation * Math.PI / 180));

                            if (pp.distanceTo(ep) < 8) {
                                score += MathHelp.getRandomNumber(10,
50);
                                explode.x = ep.x;
                                explode.y = ep.y;
                                stage.removeChild(enemys[j]);
                                stage.removeChild(plasmas[i]);
                                plasmas.splice(i--, 1);
                                enemys.splice(j--, 1);

                                var explodeClone = explode.clone();
                                explodeClone.gotoAndPlay("explode");
                                explodeClone.onAnimationEnd = function ()
{
                                    stage.removeChild(this);
                                };
                                stage.addChild(explodeClone);
                                SoundJS.play("break",
SoundJS.INTERUPT_LATE);
                            }
                        }
                    }
                }

                //处理开火
                if (shootHeld) {
                    var v = new Vector2(-Math.sin(ship.rotation * (Math.PI
/ -180)) * 10, -Math.cos(ship.rotation * (Math.PI / -180)) * 10);
                    var p = new Plasma(v,   plasmaImg);
                    p.x = ship.x;
                    p.y = ship.y;
                    p.rotation = ship.rotation;
                    plasmas.push(p);
                    stage.addChild(p);
                    SoundJS.play("laser", SoundJS.INTERUPT_LATE);
                    ship.fire();
                }

                //处理飞船的射击角度
                if (lfHeld) {
                    ship.rotation -= 2;
                }
```

```
                if (rtHeld) {
                    ship.rotation += 2;
                }


                //计算FPS
                fpsLabel.text = Math.round(Ticker.getMeasuredFPS()) + "
FPS";
            }

            //更新舞台
            stage.update();

    }
```

6.9　可玩性增强——积分、技能

积分

```
        scoreLabel = new Text("Score: --", "bold 14px Arial", "#fff");
        scoreLabel.y = 20;
        stage.addChild(scoreLabel);
```

```
    function tick() {
        if (gameStartTag) {

                //生成敌机
                if (enemys.length === 0) {
                    mineCount += 10;
                    generateMines(mineCount);
                }

                //更新积分
                scoreLabel.text = "Score: " + score;
                ....
                ....
                ....
```

```
                //处理碰撞检测
                for (var i = 0; i < plasmas.length; i++) {
                    for (var j = 0; j < enemys.length; j++) {

                        if (plasmas[i]) {
                            var ep = new Vector2(enemys[j].x + 16,
enemys[j].y + 16);
                            var pp = new Vector2(plasmas[i].x + 48 *
Math.sin(plasmas[i].rotation * Math.PI / 180), plasmas[i].y - 48 *
Math.cos(plasmas[i].rotation * Math.PI / 180));

                            if (pp.distanceTo(ep) < 8) {
                                score += 50;
                                ....
                                ....
                                ....
                            }
                        }
                    }
                }
```

飞船技能

```
    var energyHeld;
    var KEYCODE_SPACE = 32;
    function handleKeyDown(e) {

        if (!e) { var e = window.event; }
        switch (e.keyCode) {
            case KEYCODE_SPACE: energyHeld = true; break;
            ....
            ....
            ....
        }
    }

    function handleKeyUp(e) {

        if (!e) { var e = window.event; }
        switch (e.keyCode) {
            case KEYCODE_SPACE: energyHeld = false; break;
```

```
            ....
            ....
            ....
        }
    }
```

```
                if (energyHeld) {
                    for (var i = 0; i < 5; i++) {
                        var r = ship.rotation - 30 + i * 15;
                        var v = new Vector2(-Math.sin(r * (Math.PI / -180))
* 10, -Math.cos(r * (Math.PI / -180)) * 10);
                        var p = new Plasma(v, plasmaImg);
                        p.x = ship.x;
                        p.y = ship.y;
                        p.rotation = r;
                        plasmas.push(p);
                        stage.addChild(p);
                    }
                    SoundJS.play("laser", SoundJS.INTERUPT_LATE);
                    ship.fire();
                }
```