案例与和实例

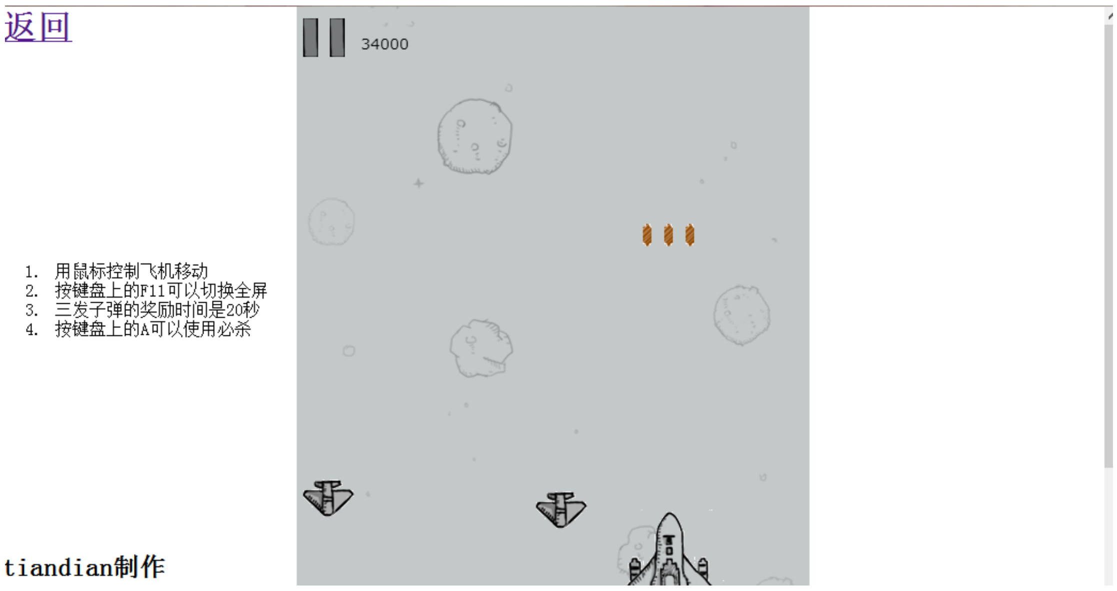
在本章节中，我们将进行有关于 HTML 5 的是实际项目案例，在学习的时候我们应该按照企业的要求严格的要求自己，务必按照相关的代码规范编写。

3.1 微信飞机大战网页版制作

我们首先分为 10 部分去不断的完善的我们的软件。在项目中我们用到的 CSS 样式为同一个资源局势 index.css

```
*{padding:0;margin:0;}
canvas{background:#ccc;}
h1{width:80px;position:absolute;top:0;left:0;}
#canvas{width:480px;height:852px;margin:0 auto;cursor: pointer;}
```

项目的实际效果图片如下：

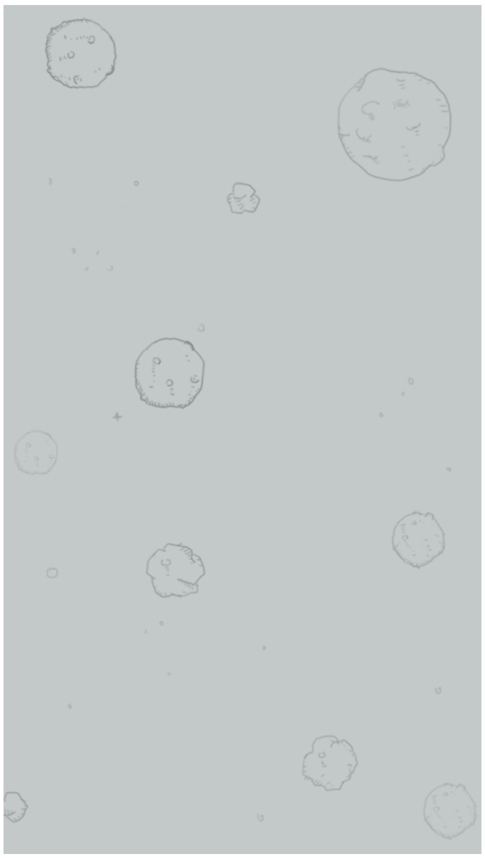


1. 用鼠标控制飞机移动
2. 按键盘上的F11可以切换全屏
3. 三发子弹的奖励时间是20秒
4. 按键盘上的A可以使用必杀

图片 游戏效果图片

3.1.1 飞行背景的滚动制作

我们用到的图片如下：



图片命名 bg

Html 01 代码如下：

```
        <!DOCTYPE html>
        <html lang="en">
        <head>
        <meta charset="utf-8" />
        <meta  http-equiv="X-UA-Compatible"  content="IE=edge,chrome=1"
/>
        <title>html5 飞机大战</title>
        <meta       name="viewport"       content="width=device-width;
initial-scale=1.0" />
        <link href="css/index.css" rel="stylesheet" />

        <script src="js/jquery.js"></script>

        <script src="js/01.js"></script>

        <!--[if lt IE 9]>
        <script src="js/html5.js"></script>
        <![endif]-->
        </head>
        <body>
        <h1><a href="./index.html">返回</a></h1>

        <div id="canvas">
        <canvas id="gameCanvas" width='480' height='852'>你的浏览器不支
持 html5，请使用谷歌、火狐、IE9 或更高级的浏览器</canvas>
        </div>
        </body>
        </html>
```

JavaScript 01 代码如下：

```
        <!DOCTYPE html>
    /**
     *画布
     */
    var canvas;
    /**
     *画笔
     */
    var paint;
    /**
     *背景图移动速度
     */
    var bgShiftY=0;
    /**
     *背景图
     */
    var bgImg;

    /**
     *清除画布
     */
    function clear() {
        paint.clearRect(0,             0,             paint.canvas.width,
paint.canvas.height);
    }

    /**
     *画场景
     */
    function drawScene() {
       clear();

         paint.drawImage(bgImg,0,bgShiftY);
         paint.drawImage(bgImg,0,bgShiftY-852);
         bgShiftY +=4;
       if (bgShiftY >=852) {
            bgShiftY =0;
        }

    }

    $(window).load(function() {
       paint=$('#gameCanvas')[0].getContext('2d');
       canvas=$('#gameCanvas');

       var width = canvas.width;
        var height = canvas.height;

        // 加载背景图片
        bgImg = new Image();
        bgImg.src = 'img/bg.png';
        bgImg.onload = function() {
        }
        bgImg.onerror = function() {
            console.log('加载背景图片出错！');
        }



        setInterval(drawScene, 30); // loop drawScene
    });
```

3.1.2 玩家飞机随鼠标移动

Html 02 代码如下：

```
    <!DOCTYPE html>
    <html lang="en">
    <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <title>html5 飞机大战</title>
    <meta name="viewport" content="width=device-width;
initial-scale=1.0" />
    <link href="css/index.css" rel="stylesheet" />

    <script src="js/jquery.js"></script>

    <script src="js/02.js"></script>

    <!--[if lt IE 9]>
    <script src="js/html5.js"></script>
    <![endif]-->
    <style>
    span{position:absolute;top:300px;right:200px;display:block;height
:100px;width:200px;}
    </style>
    </head>
    <body>
    <h1><a href="./index.html">返回</a></h1>

    <div id="canvas">
    <canvas id="gameCanvas" width='480' height='852'>你的浏览器不支持html5，请使用谷歌、火狐、IE9或更高级的浏览器</canvas>
    </div>

    <span></span>
    </body>
    </html>
```

JavaScript 01 代码如下：

```
    /**
     *画布
     */
    var canvas;
    /**
     *画笔
     */
    var paint;
    /**
     *背景图移动速度
     */
    var bgShiftY=0;
    /**
     *背景图
     */
    var bgImg;
    /**
     *玩家
     */
    var play;
    /**
     *玩家飞机宽
     */
    var playerW = 105;
    /**
     *玩家飞机高
     */
    var playerH = 128;

    var playFrame = 0; // initial sprite frame
    var iSprDir = 4; // initial dragon direction



    /**
     *获取当前的 x 坐标值
     */
    function pageX(elem){
      return
elem.offsetParent?(elem.offsetLeft+pageX(elem.offsetParent)):elem.off
setLeft;
    }

    /**
     *获取当前的 y 坐标值
     */
    function pageY(elem){
      return
elem.offsetParent?(elem.offsetTop+pageY(elem.offsetParent)):elem.offs
etTop;
    }

    /**
     *清除画布
     */
    function clear() {
        paint.clearRect(0,           0,           paint.canvas.width,
paint.canvas.height);
    }

    // Player objects
    function Player(x, y, w, h, image) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.image = image;
        this.bDrag = false;
    }
    /**
     *画场景
     */
    function drawScene() {
       clear();

         paint.drawImage(bgImg,0,bgShiftY);
         paint.drawImage(bgImg,0,bgShiftY-852);
         bgShiftY +=4;
       if (bgShiftY >=852) {
            bgShiftY =0;
        }

        playFrame++;
        if (playFrame >=2) {
            playFrame = 0;
        }

        paint.drawImage(play.image,    playFrame*play.w,0,    play.w,
play.h, play.x - play.w/2, play.y - play.h/2, play.w, play.h);


    }

    $(window).load(function() {
       paint=$('#gameCanvas')[0].getContext('2d');
```

```
        canvas=$('#gameCanvas');

        var width = canvas.width;
         var height = canvas.height;

         // 加载背景图片
         bgImg = new Image();
         bgImg.src = 'img/bg.png';
         bgImg.onload = function() {
         }
         bgImg.onerror = function() {
             console.log('加载背景图片出错！');
         }

        var playeImg = new Image();
         playeImg.src = 'img/player.png';
         playeImg.onload = function() {
         }
         play = new Player(400, 300, playerW, playerH, playeImg);

         var offLeft=pageX(canvas[0]);

         canvas.mousemove(function(e){
         $("span").text('X:'+e.pageX + ", Y:" + e.pageY);
         play.x=e.pageX-offLeft;
         play.y=e.pageY;
         });



         setInterval(drawScene, 40); // loop drawScene
     });
```

3.1.3 画出子弹

Html 03 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<title>html5 飞机大战</title>
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
<link href="css/index.css" rel="stylesheet" />
<script src="js/jquery.js"></script>
<script src="js/03.js"></script>
<!--[if lt IE 9]>
<script src="js/html5.js"></script>
<![endif]-->
<style>
span{position:absolute;top:300px;right:200px;display:block;height:100px;width:200px;}
</style>
</head>
<body>
<h1><a href="./index.html">返回</a></h1>
<div id="canvas">
<canvas id="gameCanvas" width='480' height='852'>你的浏览器不支持 html5，请使用谷歌、
火狐、IE9 或更高级的浏览器</canvas>
</div>
<span></span>
</body>
</html>
```

JavaScript 03 代码如下：

```
/**
 *画布
 */
var canvas;
/**
 *画笔
 */
var paint;
```

```
/**
 *背景图移动速度
 */
var bgShiftY=0;
/**
 *背景图
 */
var bgImg;
/**
 *玩家
 */
var play;
/**
 *玩家飞机宽
 */
var playerW = 105;
/**
 *玩家飞机高
 */
var playerH = 128;
/**
 *飞机的当前桢
 */
var playFrame = 0;
var iSprDir = 4; // initial dragon direction
/**
 *子弹数组
 */
var bullets = [];
/**
 *子弹速度
 */
var bSpeed = 10;
```

```
    var pressedKeys = []; // array of pressed keys



    /**
     *获取当前的 x 坐标值
     */
    function pageX(elem){
      return
elem.offsetParent?(elem.offsetLeft+pageX(elem.offsetParent)):elem.off
setLeft;
    }

    /**
     *获取当前的 y 坐标值
     */
    function pageY(elem){
      return
elem.offsetParent?(elem.offsetTop+pageY(elem.offsetParent)):elem.offs
etTop;
    }

    /**
     *清除画布
     */
    function clear() {
        paint.clearRect(0,            0,            paint.canvas.width,
paint.canvas.height);
    }

    /**
     *Player 对象
     */
    function Player(x, y, w, h, image) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.image = image;
        this.die = false;
    }
    /**
     *子弹  对象
     */
    function Bullet(x, y, w, h, speed, image) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.speed = speed;
        this.image = image;
    }
    /**
     *画场景
     */
    function drawScene() {
       clear();
       //背景图片滚动 start
         paint.drawImage(bgImg,0,bgShiftY);
         paint.drawImage(bgImg,0,bgShiftY-852);
         bgShiftY +=4;
        if (bgShiftY >=852) bgShiftY =0;
       // end

        //玩家飞机切帧 start
        playFrame++;
        if (playFrame >=2)playFrame = 0;
        paint.drawImage(play.image,    playFrame*play.w,0,    play.w,
play.h, play.x - play.w/2, play.y - play.h/2, play.w, play.h);
```

```
       // end

            // draw bullets
            if (bullets.length > 0) {
                for (var key in bullets) {
                    if (bullets[key] != undefined) {
                        paint.drawImage(bullets[key].image,
bullets[key].x, bullets[key].y);
                        bullets[key].y -= bullets[key].speed;

                        // if a rocket is out of screen - remove it
                        if (bullets[key].y < 0) {
                            delete bullets[key];
                        }
                    }
                }
            }

    }



    $(window).load(function() {
       paint=$('#gameCanvas')[0].getContext('2d');
       canvas=$('#gameCanvas');
       //画布宽高
       var width = canvas.width;
        var height = canvas.height;

        //画布距离浏览器左边的距离
        var offLeft=pageX(canvas[0]);
        // 加载背景图片
        bgImg = new Image();
        bgImg.src = 'img/bg.png';
```

```
         bgImg.onload = function() {
         }
         bgImg.onerror = function() {
             console.log('加载背景图片出错！');
         }

         // 加载玩家图片
        var playeImg = new Image();
         playeImg.src = 'img/player.png';
         playeImg.onload = function() {
         }
         play = new Player(240, 800, playerW, playerH, playeImg);

         // 加载子弹图片
         var bulletImg = new Image();
         bulletImg.src = 'img/bullet.png';
         bulletImg.onload = function() {

         function creatBullet(){
           bullets.push(new  Bullet(play.x  -5,  play.y  -  play.h,  32,
32,bSpeed, bulletImg))
        }
        if(!play.die)
        creat_bullet=setInterval(creatBullet, 200);
        else
        clearInterval(creat_bullet);
         }
         /*
         $(window).keyup(function (evt) { // onkeyup event handle
             var pk = pressedKeys[evt.keyCode];
             if (pk) {
                 delete pressedKeys[evt.keyCode]; // remove pressed key
from array
             }
```

```
            if (evt.keyCode == 65) { // 'A' button - add a rocket
              play.die=true;
              clearInterval(creat_bullet);
            }

        });

        */

        $("span").text('num:'+bullets.length);

        //玩家飞机跟随鼠标移动
        canvas.mousemove(function(e){
       // $("span").text('X:'+e.pageX + ", Y:" + e.pageY);
        play.x=e.pageX-offLeft;
        play.y=e.pageY;
        });



        setInterval(drawScene, 40); // loop drawScene
    });
```

3.1.4 画出敌人

Html 04 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<title>html5 飞机大战</title>
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
<link href="css/index.css" rel="stylesheet" />
<script src="js/jquery.js"></script>
```

```
<script src="js/04.js"></script>
<!--[if lt IE 9]>
<script src="js/html5.js"></script>
<![endif]-->
<style>
span{position:absolute;top:300px;right:200px;display:block;height:100px;width:200px;}
</style>
</head>
<body>
<h1><a href="./index.html">返回</a></h1>
<div id="canvas">
<canvas id="gameCanvas" width='480' height='852'>你的浏览器不支持 html5，请使用谷歌、火狐、IE9 或更高级的浏览器</canvas>
</div>
<span></span>
</body>
</html>
```

JavaScript 04 代码如下：

```
/**
 *画布
 */
var canvas;
/**
 *画笔
 */
var paint;
/**
 *背景图移动速度
 */
var bgShiftY=0;
/**
 *背景图
 */
var bgImg;
/**
```

```
 *玩家
 */
var play;
/**
 *玩家飞机宽
 */
var playerW = 105;
/**
 *玩家飞机高
 */
var playerH = 128;
/**
 *飞机的当前桢
 */
var playFrame = 0;
var iSprDir = 4; // initial dragon direction
/**
 *子弹数组
 */
var bullets = [];
/**
 *子弹速度
 */
var bSpeed = 50;

var pressedKeys = []; // array of pressed keys
/**
 *e0 宽，小型飞机
 */
var iEnemyW =48; // enemy width
/**
 *e0 高，小型飞机
 */
var iEnemyH = 37; // enemy height
```

```
    /**
     *e0 数组
     */
    var enemies = [];

    var enTimer = null; // random timer for a new enemy

    var iEnemySpeed = 5; // initial enemy speed
    /**
     *获取当前 html 元素的 x 坐标值
     */
    function pageX(elem){
      return
elem.offsetParent?(elem.offsetLeft+pageX(elem.offsetParent)):elem.off
setLeft;
    }

    /**
     *获取当前 html 元素的 y 坐标值
     */
    function pageY(elem){
      return
elem.offsetParent?(elem.offsetTop+pageY(elem.offsetParent)):elem.offs
etTop;
    }

    /**
     *清除画布
     */
    function clear() {
        paint.clearRect(0,            0,            paint.canvas.width,
paint.canvas.height);
    }
```

```
/**
 *Player 对象
 */
function Player(x, y, w, h, image) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.image = image;
    this.die = false;
}
/**
 *子弹  对象
 */
function Bullet(x, y, w, h, speed, image) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.speed = speed;
    this.image = image;
}
/**
 *敌人  对象
 */
function Enemy(x, y, w, h, speed, image) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.speed = speed;
    this.image = image;
}
```

```
    // get random number between X and Y
    function getRand(x, y) {
        return Math.floor(Math.random()*y)+x;
    }




    /**
     *画场景
     */
    function drawScene() {
       clear();
       //背景图片滚动 start
         paint.drawImage(bgImg,0,bgShiftY);
         paint.drawImage(bgImg,0,bgShiftY-852);
         bgShiftY +=4;
        if (bgShiftY >=852) bgShiftY =0;
       // end

        //玩家飞机切帧 start
        playFrame++;
        if (playFrame >=2)playFrame = 0;
        paint.drawImage(play.image,    playFrame*play.w,0,    play.w,
play.h, play.x - play.w/2, play.y - play.h/2, play.w, play.h);
       // end

            // draw bullets
            if (bullets.length > 0) {
                for (var key in bullets) {
                    if (bullets[key] != undefined) {
                        paint.drawImage(bullets[key].image,
bullets[key].x, bullets[key].y);
                        bullets[key].y -= bullets[key].speed;
                        // if a rocket is out of screen - remove it
                        if (bullets[key].y < 0) {
                            delete bullets[key];
                        }
                    }
                }
            }

            // draw enemies
            if (enemies.length > 0) {
                for (var ekey in enemies) {
                    if (enemies[ekey] != undefined) {
                     //          paint.drawImage(enemies[ekey].image,
enemies[ekey].x, enemies[ekey].y,);
       paint.drawImage(enemies[ekey].image,
0,0,enemies[ekey].w,enemies[ekey].h,enemies[ekey].x,enemies[ekey].y,e
nemies[ekey].w,enemies[ekey].h);
                        enemies[ekey].y -= enemies[ekey].speed;

                     //$("span").text('X:'+play.x + ", Y:" + play.y);
                     //$("span").text('X:'+enemies[ekey].x + ", Y:" +
enemies[ekey].y);
                        // remove an enemy object if it is out of screen
                        if (enemies[ekey].y > canvas.height) {
                            delete enemies[ekey];
                        }
                    }
                }
            }

    }
    $(window).load(function() {
       paint=$('#gameCanvas')[0].getContext('2d');
       canvas=$('#gameCanvas');
       //画布宽高
       var width = canvas.width;
        var height = canvas.height;

        //画布距离浏览器左边的距离
        var offLeft=pageX(canvas[0]);
        // 加载背景图片
        bgImg = new Image();
        bgImg.src = 'img/bg.png';
        bgImg.onload = function() {
        }
        bgImg.onerror = function() {
            console.log('加载背景图片出错！');
        }

        // 加载玩家图片
       var playeImg = new Image();
        playeImg.src = 'img/player.png';
        playeImg.onload = function() {
        }
        play = new Player(240, 800, playerW, playerH, playeImg);

        // 加载子弹图片
        var bulletImg = new Image();
        bulletImg.src = 'img/bullet.png';
        bulletImg.onload = function() {

        function creatBullet(){
           bullets.push(new  Bullet(play.x  -5,  play.y  -  play.h,  32,
32,bSpeed, bulletImg))
       }
```

```
        if(!play.die)
        creat_bullet=setInterval(creatBullet,40);
        else
        clearInterval(creat_bullet);
         }

         // initialization of empty enemy
         var e0Img = new Image();
         e0Img.src = 'img/e0.png';
         e0Img.onload = function() {

            function addEnemy() {
                clearInterval(enTimer);
               //48-432 之间
                var randX =Math.floor(Math.random()*432);
                enemies.push(new   Enemy(randX,   -iEnemyH,   iEnemyW,
iEnemyH, - iEnemySpeed, e0Img));
            // $("span").text('X:'+randX);
                var interval = getRand(100, 400);
                enTimer = setInterval(addEnemy, interval); // loop
            }
        addEnemy();
         }



         /*
         $(window).keyup(function (evt) { // onkeyup event handle
             var pk = pressedKeys[evt.keyCode];
             if (pk) {
                 delete pressedKeys[evt.keyCode]; // remove pressed key
from array
             }
             if (evt.keyCode == 65) { // 'A' button - add a rocket
               play.die=true;
```

```
                clearInterval(creat_bullet);
              }

         });

         */

        // $("span").text('num:'+bullets.length);

         //玩家飞机跟随鼠标移动
         canvas.mousemove(function(e){
        // $("span").text('X:'+e.pageX + ", Y:" + e.pageY);
         play.x=e.pageX-offLeft;
         play.y=e.pageY;
         });



         setInterval(drawScene,40); // loop drawScene

     });
```

3.1.5 碰撞检测与爆炸效果

Html 05 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<title>html5 飞机大战</title>
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
<link href="css/index.css" rel="stylesheet" />
<script src="js/jquery.js"></script>
<script src="js/05.js"></script>
```

```
<!--[if lt IE 9]>
<script src="js/html5.js"></script>
<![endif]-->
<style>
span{position:absolute;top:300px;right:200px;display:block;height:100px;width:200px;}
</style>
</head>
<body>
<h1><a href="./index.html">返回</a></h1>
<div id="canvas">
<canvas id="gameCanvas" width='480' height='852'>你的浏览器不支持 html5，请使用谷歌、火狐、IE9 或更高级的浏览器</canvas>
</div>
<span></span>
</body>
</html>
```

JavaScript 05 代码如下：

```
/**
 *画布
 */
var canvas;
/**
 *画笔
 */
var paint;
/**
 *背景图移动速度
 */
var bgShiftY=0;
/**
 *背景图
 */
var bgImg;
/**
 *玩家
 */
var play;
/**
 *玩家飞机宽
 */
var playerW = 105;
/**
 *玩家飞机高
 */
var playerH = 128;
/**
 *飞机的当前桢
 */
var playFrame = 0;
var iSprDir = 4; // initial dragon direction
/**
 *子弹数组
 */
var bullets = [];
/**
 *子弹速度
 */
var bSpeed = 50;

var pressedKeys = []; // array of pressed keys
/**
 *e0 宽，小型飞机
 */
var iEnemyW =48; // enemy width
/**
 *e0 高，小型飞机
 */
var iEnemyH = 37; // enemy height
/**
```

```
     *e0 数组
     */
    var enemies = [];

    var enTimer = null; // random timer for a new enemy

    var e1Timer = null;

    var iEnemySpeed = 5; // initial enemy speed

    var   bossTimer=null;

    var e0Frame=1;

    var bgSound; // bg sound
    /**
     *爆炸数组
     */
    var explosions = []; // array of explosions
    /**
     *获取当前 html 元素的 x 坐标值
     */
    function pageX(elem){
      return
elem.offsetParent?(elem.offsetLeft+pageX(elem.offsetParent)):elem.off
setLeft;
    }

    /**
     *获取当前 html 元素的 y 坐标值
     */
    function pageY(elem){
      return
elem.offsetParent?(elem.offsetTop+pageY(elem.offsetParent)):elem.offs
etTop;
    }

    /**
     *清除画布
     */
    function clear() {
        paint.clearRect(0,            0,            paint.canvas.width,
paint.canvas.height);
    }

    /**
     *Player 对象
     */
    function Player(x, y, w, h, image) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.image = image;
        this.die = false;
    }
    /**
     *子弹  对象
     */
    function Bullet(x, y, w, h, speed,power, image) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.power=power;
        this.speed = speed;
        this.image = image;
    }
```

```
/**
 *敌人  对象
 */
function Enemy(x, y, w, h, speed,hp, image) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.hp = hp;
    this.speed = speed;
    this.image = image;
}
/**
 *爆炸  对象
 */
function Explosion(x, y, w, h, sprite, image) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.sprite = sprite;
    this.image = image;
  //  this.frame = frame;
}
// get random number between X and Y
function getRand(x, y) {
    return Math.floor(Math.random()*y)+x;
}



/**
 *画场景
 */
```

```
    function drawScene() {
       clear();
       //背景图片滚动 start
         paint.drawImage(bgImg,0,bgShiftY);
         paint.drawImage(bgImg,0,bgShiftY-852);
         bgShiftY +=4;
        if (bgShiftY >=852) bgShiftY =0;
       // end

        //玩家飞机切帧 start
        playFrame++;
        if (playFrame >=2)playFrame = 0;
        paint.drawImage(play.image,    playFrame*play.w,0,    play.w,
play.h, play.x - play.w/2, play.y - play.h/2, play.w, play.h);
       // end

            // draw bullets
            if (bullets.length > 0) {
                for (var key in bullets) {
                    if (bullets[key] != undefined) {
                        paint.drawImage(bullets[key].image,
bullets[key].x, bullets[key].y);
                        bullets[key].y -= bullets[key].speed;

                        // if a rocket is out of screen - remove it
                        if (bullets[key].y < 0) {
                            delete bullets[key];
                        }
                    }
                }
            }



            // draw explosions
            if (explosions.length > 0) {
                for (var key in explosions) {
                    if (explosions[key] != undefined) {
                        // display explosion sprites
                        paint.drawImage(explosions[key].image,
explosions[key].sprite*explosions[key].w,    0,    explosions[key].w,
explosions[key].h,    explosions[key].x    -    explosions[key].w/2,
explosions[key].y    -    explosions[key].h/2,    explosions[key].w,
explosions[key].h);
                        explosions[key].sprite++;

                        // remove an explosion object when it expires
                        if (explosions[key].sprite >6) {
                            delete explosions[key];
                        }
                    }
                }
            }

            // draw enemies
            if (enemies.length > 0) {
                for (var ekey in enemies) {
                    if (enemies[ekey] != undefined) {

       paint.drawImage(enemies[ekey].image,
0,0,enemies[ekey].w,enemies[ekey].h,enemies[ekey].x,enemies[ekey].y,e
nemies[ekey].w,enemies[ekey].h);
                        enemies[ekey].y -= enemies[ekey].speed;

                        // remove an enemy object if it is out of screen
                        if (enemies[ekey].y > canvas.height) {
                            delete enemies[ekey];
                        }
                    }
                }
            }

            if (enemies.length > 0) {
                for (var ekey in enemies) {
                    if (enemies[ekey] != undefined) {
                        // collisions with bullets
                        if (bullets.length > 0) {
                            for (var key in bullets) {
                                if          (bullets[key]          !=
undefined&&enemies[ekey] != undefined) {
                                  //       if    (bullets[key].y    <
(enemies[ekey].y    +    enemies[ekey].h/2)    &&    bullets[key].x    >
enemies[ekey].x && (bullets[key].x + bullets[key].w )< (enemies[ekey].x
+ enemies[ekey].w)) {

if(Math.pow((bullets[key].y-(enemies[ekey].y+enemies[ekey].h/2) ),2)+
Math.pow((bullets[key].x-(enemies[ekey].x+enemies[ekey].w/2)),2)<Math
.pow(enemies[ekey].w/2,2)){
       enemies[ekey].hp-=bullets[key].power;

       //$("span").text('enemies:HP='+enemies[ekey].hp);

       if(enemies[ekey].hp<=0){
           enemies[ekey].speed=0;
       explosions.push(new Explosion(enemies[ekey].x + enemies[ekey].w
/      2,      enemies[ekey].y      +      enemies[ekey].h      /      2,
enemies[ekey].w,enemies[ekey].h, 0, enemies[ekey].image));
                                      delete enemies[ekey];
       }
       delete bullets[key];
                                     //  iScore++;
                                 }
                             }
                            }
                        }


                    }
                }
            }

    }



    $(window).load(function() {
       paint=$('#gameCanvas')[0].getContext('2d');
       canvas=$('#gameCanvas');
       //画布宽高
       var width = canvas.width;
        var height = canvas.height;




        //画布距离浏览器左边的距离
        var offLeft=pageX(canvas[0]);
        // 加载背景图片
        bgImg = new Image();
        bgImg.src = 'img/bg.png';
        bgImg.onload = function() {
        }
        bgImg.onerror = function() {
            console.log('加载背景图片出错！');
        }

        // 加载玩家图片
       var playeImg = new Image();
        playeImg.src = 'img/player.png';
        playeImg.onload = function() {
        }
        play = new Player(240, 800, playerW, playerH, playeImg);

        // 加载子弹图片
        var bulletImg = new Image();
        bulletImg.src = 'img/bullet.png';
        bulletImg.onload = function() {

        function creatBullet(){
           bullets.push(new  Bullet(play.x  -5,  play.y  -  play.h,  32,
32,bSpeed,20,bulletImg))
       }
       if(!play.die)
       creat_bullet=setInterval(creatBullet,200);
       else
       clearInterval(creat_bullet);
        }

        // initialization of empty enemy
        var e0Img = new Image();
        e0Img.src = 'img/e0.png';
        e0Img.onload = function() {

           function addEnemy() {
                clearInterval(enTimer);
               iEnemySpeed=getRand(5,10);
              //48-432 之间
              //                        var                          randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
                var randX =Math.floor(Math.random()*432);
                enemies.push(new    Enemy(randX,    -iEnemyH,    iEnemyW,
iEnemyH, - iEnemySpeed,20,e0Img));
          // $("span").text('X:'+randX);
              var interval = getRand(100, 400);
              enTimer = setInterval(addEnemy, interval); // loop
          }
       addEnemy();
        }



        var e1Img = new Image();
        e1Img.src = 'img/e1.png';
        e1Img.onload = function() {

          function addEnemy() {
              clearInterval(e1Timer);
             iEnemySpeed=getRand(5,10);
             //48-432 之间
             //                     var                        randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
              var randX =Math.floor(Math.random()*432);
              enemies.push(new   Enemy(randX,   -68,   68,   94,   -
iEnemySpeed,60,e1Img));
          // $("span").text('X:'+randX);
              var interval = getRand(1000, 4000);
              e1Timer = setInterval(addEnemy, interval); // loop
          }
       addEnemy();
        }

        var bossImg = new Image();
        bossImg.src = 'img/boss.png';
        bossImg.onload = function() {

          function addEnemy() {
               clearInterval(bossTimer);
              iEnemySpeed=getRand(5,10);
             //48-432 之间
             //                        var                        randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
              var randX =Math.floor(Math.random()*332);
              enemies.push(new    Enemy(randX,    -257,172,257,     -
iEnemySpeed,100,bossImg));
          //  $("span").text('X:'+randX);
              var interval = getRand(4000, 8000);
              bossTimer = setInterval(addEnemy, interval); // loop
          }
       addEnemy();
        }

        /*
        $(window).keyup(function (evt) { // onkeyup event handle
            var pk = pressedKeys[evt.keyCode];
            if (pk) {
                delete pressedKeys[evt.keyCode]; // remove pressed key
from array
            }
            if (evt.keyCode == 65) { // 'A' button - add a rocket
              play.die=true;
              clearInterval(creat_bullet);
            }

        });

        */

       // $("span").text('num:'+bullets.length);
            // 'bg' music init
        bgSound = new Audio('media/wj.wav');
        bgSound.volume = 0.9;
        bgSound.addEventListener('ended', function() { // looping bg
sound
            this.currentTime = 0;
            this.play();
        }, false);
        bgSound.play();

        //玩家飞机跟随鼠标移动
        canvas.mousemove(function(e){
       // $("span").text('X:'+e.pageX + ", Y:" + e.pageY);
        play.x=e.pageX-offLeft;
        play.y=e.pageY;
        });



        setInterval(drawScene,40); // loop drawScene

    });
```

3.1.6 爆炸切帧改进

Html 06 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<title>html5 飞机大战</title>
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
<link href="css/index.css" rel="stylesheet" />
<script src="js/jquery.js"></script>
<script src="js/06.js"></script>
```

```
<!--[if lt IE 9]>
<script src="js/html5.js"></script>
<![endif]-->
<style>
span{position:absolute;top:300px;right:200px;display:block;height:100px;width:200px;}
</style>
</head>
<body>
<h1><a href="./index.html">返回</a></h1>
<div id="canvas">
<canvas id="gameCanvas" width='480' height='852'>你的浏览器不支持 html5，请使用谷歌、
火狐、IE9 或更高级的浏览器</canvas>
</div>
<span></span>
</body>
</html>
```

JavaScript 06 代码如下：

```
/**
 *画布
 */
var canvas;
/**
 *画笔
 */
var paint;
/**
 *背景图移动速度
 */
var bgShiftY=0;
/**
 *背景图
 */
var bgImg;
/**
```

```
 *玩家
 */
var play;
/**
 *玩家飞机宽
 */
var playerW = 105;
/**
 *玩家飞机高
 */
var playerH = 128;
/**
 *飞机的当前桢
 */
var playFrame = 0;
var iSprDir = 4; // initial dragon direction
/**
 *子弹数组
 */
var bullets = [];
/**
 *子弹速度
 */
var bSpeed = 50;

var pressedKeys = []; // array of pressed keys
/**
 *e0 宽，小型飞机
 */
var iEnemyW =48; // enemy width
/**
 *e0 高，小型飞机
 */
var iEnemyH = 37; // enemy height
```

```
    /**
     *e0 数组
     */
    var enemies = [];

    var enTimer = null; // random timer for a new enemy

    var e1Timer = null;

    var iEnemySpeed = 5; // initial enemy speed

    var   bossTimer=null;

    var e0Frame=1;

    var curFrame=0;
    var curFramee=0;
    var bgSound; // bg sound
    /**
     *爆炸数组
     */
    var explosions = []; // array of explosions
    /**
     *获取当前 html 元素的 x 坐标值
     */
    function pageX(elem){
      return
elem.offsetParent?(elem.offsetLeft+pageX(elem.offsetParent)):elem.off
setLeft;
    }

    /**
     *获取当前 html 元素的 y 坐标值
     */
    function pageY(elem){
      return
elem.offsetParent?(elem.offsetTop+pageY(elem.offsetParent)):elem.offs
etTop;
    }

    /**
     *清除画布
     */
    function clear() {
        paint.clearRect(0,            0,            paint.canvas.width,
paint.canvas.height);
    }

    /**
     *Player 对象
     */
    function Player(x, y, w, h, image) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.image = image;
        this.die = false;
    }
    /**
     *子弹  对象
     */
    function Bullet(x, y, w, h, speed,power, image) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.power=power;
        this.speed = speed;
        this.image = image;
    }
    /**
     *敌人  对象
     */
    function           Enemy(x,           y,           w,           h,
speed,hp,image,changeFrame,totleFrame,count) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.hp = hp;
        this.cf = changeFrame;//要显示的桢
        this.tf = totleFrame;//总的桢数
       this.count = count;//计算用
        this.speed = speed;
        this.image = image;
    }
    /**
     *爆炸  对象
     */
    function Explosion(x, y, w, h, sprite, image,frame) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.sprite = sprite;
        this.image = image;
       this.f = frame;//飞机爆炸的桢数
    }
    // get random number between X and Y
    function getRand(x, y) {
        return Math.floor(Math.random()*y)+x;
    }




    /**
     *画场景
     */
    function drawScene() {
       //clear();
       //背景图片滚动 start
         paint.drawImage(bgImg,0,bgShiftY);
         paint.drawImage(bgImg,0,bgShiftY-852);
         bgShiftY +=4;
        if (bgShiftY >=852) bgShiftY =0;
       // end

        //玩家飞机切帧 start
        playFrame++;
        if (playFrame >=2)playFrame = 0;
        paint.drawImage(play.image,    playFrame*play.w,0,    play.w,
play.h, play.x - play.w/2, play.y - play.h/2, play.w, play.h);
       // end

            // draw bullets
            if (bullets.length > 0) {
                for (var key in bullets) {
                    if (bullets[key] != undefined) {
                        paint.drawImage(bullets[key].image,
bullets[key].x, bullets[key].y);
                        bullets[key].y -= bullets[key].speed;

                        // if a rocket is out of screen - remove it
                        if (bullets[key].y < 0) {
                            delete bullets[key];
                        }
                    }
                }
            }

            // draw explosions
            if (explosions.length > 0) {
                for (var key in explosions) {
                    if (explosions[key] != undefined) {
                        // display explosion sprites

                        paint.drawImage(explosions[key].image,
explosions[key].sprite*explosions[key].w,    0,    explosions[key].w,
explosions[key].h,    explosions[key].x    -    explosions[key].w/2,
explosions[key].y    -    explosions[key].h/2,    explosions[key].w,
explosions[key].h);
                        explosions[key].sprite++;

                        // remove an explosion object when it expires
                        if (explosions[key].sprite >explosions[key].f)
{
                            delete explosions[key];
                        }
                    }
                }
            }

            // draw enemies
            if (enemies.length > 0) {

            //   curFramee++;
           //    $('span').text('curFramee='+curFramee);
                for (var ekey in enemies) {
                    if (enemies[ekey] != undefined) {
                 //  var curFramee=0;
                     enemies[ekey].y -= enemies[ekey].speed;
                     //if(curFramee>=enemies[ekey].cf)curFramee=0;

       paint.drawImage(enemies[ekey].image,enemies[ekey].count*(enemi
es[ekey].w),0,enemies[ekey].w,enemies[ekey].h,enemies[ekey].x,enemies
[ekey].y,enemies[ekey].w,enemies[ekey].h);

        enemies[ekey].count++;
                     if(enemies[ekey].count>=
enemies[ekey].cf)enemies[ekey].count=0;



                     //$('span').text('enemies[ekey].y
='+enemies[ekey].y );
                        // remove an enemy object if it is out of screen
                        if (enemies[ekey].y > canvas.height) {
                            delete enemies[ekey];
                        }
                    }
                }
            }

            if (enemies.length > 0) {
                for (var ekey in enemies) {
                    if (enemies[ekey] != undefined) {
                        // collisions with bullets
                        if (bullets.length > 0) {
                            for (var key in bullets) {
                                if          (bullets[key]          !=
undefined&&enemies[ekey] != undefined) {
                                  //        if    (bullets[key].y    <
(enemies[ekey].y    +    enemies[ekey].h/2)    &&    bullets[key].x    >
enemies[ekey].x && (bullets[key].x + bullets[key].w )< (enemies[ekey].x
+ enemies[ekey].w)) {

if(Math.pow((bullets[key].y-(enemies[ekey].y+enemies[ekey].h/2) ),2)+
Math.pow((bullets[key].x-(enemies[ekey].x+enemies[ekey].w/2)),2)<Math
.pow(enemies[ekey].w/2,2)){
        enemies[ekey].hp-=bullets[key].power;

        curFrame++;
        if(curFrame>enemies[ekey].cf)curFrame=0;
        paint.drawImage(enemies[ekey].image,
curFrame*enemies[ekey].w,0,enemies[ekey].w,enemies[ekey].h,enemies[ek
ey].x,enemies[ekey].y,enemies[ekey].w,enemies[ekey].h);

        //$("span").text('enemies:HP='+enemies[ekey].hp);

        if(enemies[ekey].hp<=0){
            enemies[ekey].speed=0;
        explosions.push(new Explosion(enemies[ekey].x + enemies[ekey].w
/      2,      enemies[ekey].y      +      enemies[ekey].h      /      2,
enemies[ekey].w,enemies[ekey].h,                                       0,
enemies[ekey].image,enemies[ekey].tf));
                                            delete enemies[ekey];
        }
        delete bullets[key];
                                           //  iScore++;
                                         }
                                    }
                               }
                          }



                     }
```

```
                }
            }

    }



    $(window).load(function() {
       paint=$('#gameCanvas')[0].getContext('2d');
       canvas=$('#gameCanvas');
       //画布宽高
       var width = canvas.width;
        var height = canvas.height;




        //画布距离浏览器左边的距离
        var offLeft=pageX(canvas[0]);
        // 加载背景图片
        bgImg = new Image();
        bgImg.src = 'img/bg.png';
        bgImg.onload = function() {
        }
        bgImg.onerror = function() {
            console.log('加载背景图片出错！');
        }

        // 加载玩家图片
       var playeImg = new Image();
        playeImg.src = 'img/player.png';
        playeImg.onload = function() {
        }
        play = new Player(240, 800, playerW, playerH, playeImg);
        // 加载子弹图片
        var bulletImg = new Image();
        bulletImg.src = 'img/bullet.png';
        bulletImg.onload = function() {

        function creatBullet(){
          bullets.push(new  Bullet(play.x  -5,  play.y  -  play.h,  32,
32,bSpeed,20,bulletImg))
       }
       if(!play.die)
       creat_bullet=setInterval(creatBullet,200);
       else
       clearInterval(creat_bullet);
        }

        // initialization of empty enemy
        var e0Img = new Image();
        e0Img.src = 'img/e0.png';
        e0Img.onload = function() {

          function addEnemy() {
              clearInterval(enTimer);
              iEnemySpeed=getRand(4,10);
             //48-432 之间
             //                    var                          randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
              var randX =Math.floor(Math.random()*432);
              enemies.push(new   Enemy(randX,   -iEnemyH,   iEnemyW,
iEnemyH, - iEnemySpeed,20,e0Img,0,4,0));
          //  $("span").text('X:'+randX);
              var interval = getRand(200, 400);
              enTimer = setInterval(addEnemy, interval); // loop
          }
        addEnemy();
         }



         var e1Img = new Image();
         e1Img.src = 'img/el.png';
         e1Img.onload = function() {

            function addEnemy() {
                clearInterval(e1Timer);
               iEnemySpeed=getRand(5,5);
               //48-432 之间
               //                        var                        randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
                var randX =Math.floor(Math.random()*432);
                enemies.push(new   Enemy(randX,   -68,   68,   94,   -
iEnemySpeed,60,e1Img,1,5,0));
            // $("span").text('X:'+randX);
                var interval = getRand(1000, 4000);
                e1Timer = setInterval(addEnemy, interval); // loop
            }
        addEnemy();
         }

         var bossImg = new Image();
         bossImg.src = 'img/boss2.png';
         bossImg.onload = function() {

            function addEnemy() {
                clearInterval(bossTimer);
               iEnemySpeed=getRand(5,5);
               //48-432 之间
               //                        var                        randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
```

```
               var randX =Math.floor(Math.random()*332);
               enemies.push(new    Enemy(randX,    -257,172,257,    -
iEnemySpeed,200,bossImg,2,10,0));
           //  $("span").text('X:'+randX);
               var interval = getRand(4000, 8000);
               bossTimer = setInterval(addEnemy, interval); // loop
           }
       addEnemy();
        }

        /*
        $(window).keyup(function (evt) { // onkeyup event handle
            var pk = pressedKeys[evt.keyCode];
            if (pk) {
                delete pressedKeys[evt.keyCode]; // remove pressed key
from array
            }
            if (evt.keyCode == 65) { // 'A' button - add a rocket
              play.die=true;
              clearInterval(creat_bullet);
            }

        });

        */

       // $("span").text('num:'+bullets.length);
            // 'bg' music init

        // bgSound = new Audio('media/wj.wav');
        // bgSound.volume = 0.9;
        // bgSound.addEventListener('ended', function() { // looping bg
sound
            // this.currentTime = 0;
            // this.play();
        // }, false);
        // bgSound.play();

        //玩家飞机跟随鼠标移动
        canvas.mousemove(function(e){
       // $("span").text('X:'+e.pageX + ", Y:" + e.pageY);
        play.x=e.pageX-offLeft;
        play.y=e.pageY;
        });



        setInterval(drawScene,30); // loop drawScene

    });
```

3.1.7 分数的计算

Html 07 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<title>html5 飞机大战</title>
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
<link href="css/index.css" rel="stylesheet" />
<script src="js/jquery.js"></script>
<script src="js/07.js"></script>
<!--[if lt IE 9]>
<script src="js/html5.js"></script>
<![endif]-->
<style>
span{position:absolute;top:300px;right:200px;display:block;height:100px;width:200px;}
</style>
```

```
</head>
<body>
<h1><a href="./index.html">返回</a></h1>
<div id="canvas">
<canvas id="gameCanvas" width='480' height='852'>你的浏览器不支持 html5，请使用谷歌、火狐、IE9 或更高级的浏览器</canvas>
</div>
<span></span>
</body>
</html>
```

JavaScript 07 代码如下：

```
/**
 *画布
 */
var canvas;
/**
 *画笔
 */
var paint;
/**
 *背景图移动速度
 */
var bgShiftY=0;
/**
 *背景图
 */
var bgImg;
/**
 *玩家
 */
var play;
/**
 *玩家飞机宽
 */
var playerW = 105;
```

```
/**
 *玩家飞机高
 */
var playerH = 128;
/**
 *飞机的当前桢
 */
var playFrame = 0;
var iSprDir = 4; // initial dragon direction
/**
 *子弹数组
 */
var bullets = [];
/**
 *子弹速度
 */
var bSpeed = 50;

var pressedKeys = []; // array of pressed keys
/**
 *e0 宽，小型飞机
 */
var iEnemyW =48; // enemy width
/**
 *e0 高，小型飞机
 */
var iEnemyH = 37; // enemy height
/**
 *e0 数组
 */
var enemies = [];

var enTimer = null; // random timer for a new enemy
```

```
    var e1Timer = null;

    var iEnemySpeed = 5; // initial enemy speed

    var   bossTimer=null;

    var e0Frame=1;

    var curFrame=0;
    var curFramee=0;
    var bgSound; // bg sound

    var iScore=0;

    var iScore = 0; // total score
    var iLife =50; // total life of play

    var die =false; // game pause
    var press=false;
    /**
     *爆炸数组
     */
    var explosions = []; // array of explosions
    /**
     *获取当前 html 元素的 x 坐标值
     */
    function pageX(elem){
      return
elem.offsetParent?(elem.offsetLeft+pageX(elem.offsetParent)):elem.off
setLeft;
    }

    /**
     *获取当前 html 元素的 y 坐标值
     */
    function pageY(elem){
      return
elem.offsetParent?(elem.offsetTop+pageY(elem.offsetParent)):elem.offs
etTop;
    }

    /**
     *清除画布
     */
    function clear() {
        paint.clearRect(0,            0,            paint.canvas.width,
paint.canvas.height);
    }

    /**
     *Player 对象
     */
    function Player(x, y, w, h, image,img_e) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.image = image;
        this.e=img_e;
        this.die = false;
    }
    /**
     *子弹  对象
     */
    function Bullet(x, y, w, h, speed,power, image) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.power=power;
        this.speed = speed;
        this.image = image;
    }
    /**
     *敌人  对象
     */
    function          Enemy(x,          y,          w,          h,
speed,hp,image,changeFrame,totleFrame,count,score) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.hp = hp;
        this.cf = changeFrame;//要显示的桢
        this.tf = totleFrame;//总的桢数
       this.count = count;//计算用
        this.speed = speed;
        this.image = image;
        this.score = score;
    }
    /**
     *爆炸  对象
     */
    function Explosion(x, y, w, h, sprite, image,frame) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.sprite = sprite;
        this.image = image;
       this.f = frame;//飞机爆炸的桢数
    }
    // get random number between X and Y
    function getRand(x, y) {
        return Math.floor(Math.random()*y)+x;
    }





    /**
     *画场景
     */
    function drawScene() {
       if(!die){



       clear();
       //背景图片滚动 start
         paint.drawImage(bgImg,0,bgShiftY);
         paint.drawImage(bgImg,0,bgShiftY-852);
         bgShiftY +=4;
        if (bgShiftY >=852) bgShiftY =0;
       // end

        //玩家飞机切帧 start
        playFrame++;
        if (playFrame >=2)playFrame = 0;
        paint.drawImage(play.image,    playFrame*play.w,0,    play.w,
play.h, play.x - play.w/2, play.y - play.h/2, play.w, play.h);
       // end

             // draw bullets
             if (bullets.length > 0) {
                 for (var key in bullets) {
                     if (bullets[key] != undefined) {
                         paint.drawImage(bullets[key].image,
bullets[key].x, bullets[key].y);
                         bullets[key].y -= bullets[key].speed;

                         // if a rocket is out of screen - remove it
                         if (bullets[key].y < 0) {
                             delete bullets[key];
                         }
                     }
                 }
             }

             // draw explosions
             if (explosions.length > 0) {
                 for (var key in explosions) {
                     if (explosions[key] != undefined) {
                         // display explosion sprites

                         paint.drawImage(explosions[key].image,
explosions[key].sprite*explosions[key].w,    0,    explosions[key].w,
explosions[key].h,    explosions[key].x    -    explosions[key].w/2,
explosions[key].y    -    explosions[key].h/2,    explosions[key].w,
explosions[key].h);
                         explosions[key].sprite++;

                         // remove an explosion object when it expires
                         if (explosions[key].sprite >explosions[key].f)
{
                             delete explosions[key];
                         }
                     }
                 }
             }
             // draw enemies
             if (enemies.length > 0) {
                 for (var ekey in enemies) {
                     if (enemies[ekey] != undefined) {
                      enemies[ekey].y -= enemies[ekey].speed;

        paint.drawImage(enemies[ekey].image,enemies[ekey].count*(enemi
es[ekey].w),0,enemies[ekey].w,enemies[ekey].h,enemies[ekey].x,enemies
[ekey].y,enemies[ekey].w,enemies[ekey].h);
                         enemies[ekey].count++;
                      if(enemies[ekey].count>=
enemies[ekey].cf)enemies[ekey].count=0;
                         // remove an enemy object if it is out of screen
                         if (enemies[ekey].y > canvas.height) {
                             delete enemies[ekey];
                         }
                     }
                 }
             }

             if (enemies.length > 0) {
                 for (var ekey in enemies) {
                     if (enemies[ekey] != undefined) {
                         // collisions with bullets
                         if (bullets.length > 0) {
                             for (var key in bullets) {
                                 if          (bullets[key]          !=
undefined&&enemies[ekey] != undefined) {

if(Math.pow((bullets[key].y-(enemies[ekey].y+enemies[ekey].h/2) ),2)+
Math.pow((bullets[key].x-(enemies[ekey].x+enemies[ekey].w/2)),2)<Math
.pow(enemies[ekey].w/2,2)){
        enemies[ekey].hp-=bullets[key].power;
```

```
        curFrame++;
        if(curFrame>enemies[ekey].cf)curFrame=0;
        paint.drawImage(enemies[ekey].image,
curFrame*enemies[ekey].w,0,enemies[ekey].w,enemies[ekey].h,enemies[ek
ey].x,enemies[ekey].y,enemies[ekey].w,enemies[ekey].h);

        if(enemies[ekey].hp<=0){
            enemies[ekey].speed=0;
        explosions.push(new Explosion(enemies[ekey].x + enemies[ekey].w
/      2,      enemies[ekey].y      +      enemies[ekey].h      /      2,
enemies[ekey].w,enemies[ekey].h,                                      0,
enemies[ekey].image,enemies[ekey].tf));
                                            iScore+=enemies[ekey].score;
                                            delete enemies[ekey];

        }
        delete bullets[key];

                                        }
                                    }
                                }
                            }



                    // collisions with play
              if (enemies[ekey] != undefined) {
                  if
(Math.pow((play.y-(enemies[ekey].y+enemies[ekey].h)),2)+Math.pow((pla
y.x-(enemies[ekey].x+enemies[ekey].w/2)),2)<Math.pow(play.w/2,2)) {



                    // canvas.unbind('mousemove');
                       // delete enemy and make damage
                    //  delete play;
                    // play = new Player(240, 800, playerW, playerH,
playeImg,peImg);

                     iLife -= 1;
    //
                     if (iLife <= 0) { // Game over
    die = true;//
                        // draw score
                        canvas.unbind('mousemove');
                        paint.font = '14px Verdana';
                        paint.fillStyle = '#000';
                        paint.fillText('Game over, your score: ' +
iScore + ' points', 25, 200);
                        return;
                    }

                    delete play;
                    explosions.push(new Explosion(play.x , play.y ,
play.w, play.h, 0,play.e,10));


                }
            }



                    }
                }
            }





           paint.font = '14px Verdana';
```

```
        paint.fillStyle = '#000';
        paint.fillText('Life: ' + iLife , 50, 660);
        paint.fillText('Score: ' + iScore, 50, 50);
}
}



$(window).load(function() {
   paint=$('#gameCanvas')[0].getContext('2d');
   canvas=$('#gameCanvas');
   //画布宽高
   var width = canvas.width;
    var height = canvas.height;

   // getContext


    //画布距离浏览器左边的距离
    var offLeft=pageX(canvas[0]);
    // 加载背景图片
    bgImg = new Image();
    bgImg.src = 'img/bg.png';
    bgImg.onload = function() {
    }
    bgImg.onerror = function() {
        console.log('加载背景图片出错！');
    }

    // 加载玩家图片
   var playeImg = new Image();
    playeImg.src = 'img/player.png';
    playeImg.onload = function() {
    }
```

```
         // 加载玩家爆炸图片
        var peImg = new Image();
         peImg.src = 'img/p_e.png';
         peImg.onload = function() {
         }
         play = new Player(240, 800, playerW, playerH, playeImg,peImg);


         // 加载子弹图片
         var bulletImg = new Image();
         bulletImg.src = 'img/bullet.png';
         bulletImg.onload = function() {

         function creatBullet(){
            bullets.push(new  Bullet(play.x  -5,  play.y  -  play.h,  32,
32,bSpeed,20,bulletImg))
        }
        if(!play.die)
        creat_bullet=setInterval(creatBullet,200);
        else
        clearInterval(creat_bullet);
         }

         // initialization of empty enemy
         var e0Img = new Image();
         e0Img.src = 'img/e0.png';
         e0Img.onload = function() {

            function addEnemy() {
                clearInterval(enTimer);
               iEnemySpeed=getRand(4,10);
              //48-432 之间
              //                        var                        randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
```

```
                var randX =Math.floor(Math.random()*432);
                enemies.push(new   Enemy(randX,   -iEnemyH,   iEnemyW,
iEnemyH, - iEnemySpeed,20,e0Img,0,4,0,1000));
            // $("span").text('X:'+randX);
                var interval = getRand(200, 400);
                enTimer = setInterval(addEnemy, interval); // loop
            }
        addEnemy();
         }



         var e1Img = new Image();
         e1Img.src = 'img/e1.png';
         e1Img.onload = function() {

            function addEnemy() {
                clearInterval(e1Timer);
               iEnemySpeed=getRand(5,5);
              //48-432 之间
              //                      var                       randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
                var randX =Math.floor(Math.random()*432);
                enemies.push(new   Enemy(randX,   -68,   68,   94,   -
iEnemySpeed,60,e1Img,1,5,0,5000));
            // $("span").text('X:'+randX);
                var interval = getRand(1000, 4000);
                e1Timer = setInterval(addEnemy, interval); // loop
            }
        addEnemy();
         }

         var bossImg = new Image();
         bossImg.src = 'img/boss2.png';
         bossImg.onload = function() {
            function addEnemy() {
                clearInterval(bossTimer);
               iEnemySpeed=getRand(5,5);
              //48-432 之间
              //                        var                          randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
                var randX =Math.floor(Math.random()*332);
                enemies.push(new    Enemy(randX,     -257,172,257,      -
iEnemySpeed,200,bossImg,2,10,0,30000));
            //  $("span").text('X:'+randX);
                var interval = getRand(4000, 8000);
                bossTimer = setInterval(addEnemy, interval); // loop
            }
        addEnemy();
         }

         /*
         $(window).keyup(function (evt) { // onkeyup event handle
             var pk = pressedKeys[evt.keyCode];
             if (pk) {
                 delete pressedKeys[evt.keyCode]; // remove pressed key
from array
             }
             if (evt.keyCode == 65) { // 'A' button - add a rocket
               play.die=true;
               clearInterval(creat_bullet);
             }

         });

         */

        // $("span").text('num:'+bullets.length);
```

```
            // 'bg' music init

        // bgSound = new Audio('media/wj.wav');
        // bgSound.volume = 0.9;
        // bgSound.addEventListener('ended', function() { // looping bg
sound
            // this.currentTime = 0;
            // this.play();
        // }, false);
        // bgSound.play();

        //玩家飞机跟随鼠标移动

          canvas.mousemove(function(e){
       // $("span").text('X:'+e.pageX + ", Y:" + e.pageY);
        play.x=e.pageX-offLeft;
        play.y=e.pageY;
        if(!press)return;
        });





        setInterval(drawScene,30); // loop drawScene

    });
```

3.1.8 双发子弹

Html 08 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<title>html5 飞机大战</title>
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
<link href="css/index.css" rel="stylesheet" />
<script src="js/jquery.js"></script>
<script src="js/08.js"></script>
<!--[if lt IE 9]>
<script src="js/html5.js"></script>
<![endif]-->
<style>
span{position:absolute;top:300px;right:200px;display:block;height:100px;width:200px;}
</style>
</head>
<body>
<h1><a href="./index.html">返回</a></h1>
<div id="canvas">
<canvas id="gameCanvas" width='480' height='852'>你的浏览器不支持 html5，请使用谷歌、
火狐、IE9 或更高级的浏览器</canvas>
</div>
<span></span>
</body>
</html>
```

JavaScript 08 代码如下：

```
/**
 *画布
 */
var canvas;
/**
 *画笔
 */
var paint;
/**
 *背景图移动速度
 */
var bgShiftY=0;
/**
```

```
 *背景图
 */
var bgImg;
/**
 *玩家
 */
var play;
/**
 *玩家飞机宽
 */
var playerW = 105;
/**
 *玩家飞机高
 */
var playerH = 128;
/**
 *飞机的当前桢
 */
var playFrame = 0;
var iSprDir = 4; // initial dragon direction
/**
 *子弹数组
 */
var bullets = [];
/**
 *子弹速度
 */
var speedY = 50;

var pressedKeys = []; // array of pressed keys



/**
 *获取当前的 x 坐标值
```

```
     */
    function pageX(elem){
      return
elem.offsetParent?(elem.offsetLeft+pageX(elem.offsetParent)):elem.off
setLeft;
    }

    /**
     *获取当前的 y 坐标值
     */
    function pageY(elem){
      return
elem.offsetParent?(elem.offsetTop+pageY(elem.offsetParent)):elem.offs
etTop;
    }

    /**
     *清除画布
     */
    function clear() {
        paint.clearRect(0,              0,              paint.canvas.width,
paint.canvas.height);
    }

    /**
     *Player 对象
     */
    function Player(x, y, w, h, image,num) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.num=num;
        this.image = image;
        this.die = false;
    }
    /**
     *子弹  对象
     */
    function Bullet(x, y, w, h,speedx,speedy,image) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
      //  tiis.num=num;
        this.speedx = speedx;
        this.speedy = speedy;
        this.image = image;
    }
    /**
     *画场景
     */
    function drawScene() {
       clear();
       //背景图片滚动 start
         paint.drawImage(bgImg,0,bgShiftY);
         paint.drawImage(bgImg,0,bgShiftY-852);
         bgShiftY +=4;
        if (bgShiftY >=852) bgShiftY =0;
       // end

        //玩家飞机切帧 start
        playFrame++;
        if (playFrame >=2)playFrame = 0;
        paint.drawImage(play.image,    playFrame*play.w,0,    play.w,
play.h, play.x - play.w/2, play.y - play.h/2, play.w, play.h);
       // end
```

```
            // draw bullets
            if (bullets.length > 0) {
                for (var key in bullets) {
                    if (bullets[key] != undefined) {
       paint.drawImage(bullets[key].image,            bullets[key].x,
bullets[key].y);
                       bullets[key].y -= bullets[key].speedy;
                       bullets[key].x += bullets[key].speedx;
                        // if a rocket is out of screen - remove it
                        if (bullets[key].y < 0) {
                            delete bullets[key];
                        }
                    }
                }
            }

    }



    $(window).load(function() {
       paint=$('#gameCanvas')[0].getContext('2d');
       canvas=$('#gameCanvas');
       //画布宽高
       var width = canvas.width;
        var height = canvas.height;

        //画布距离浏览器左边的距离
        var offLeft=pageX(canvas[0]);
        // 加载背景图片
        bgImg = new Image();
        bgImg.src = 'img/bg.png';
        bgImg.onload = function() {
        }
```

```
         bgImg.onerror = function() {
             console.log('加载背景图片出错！');
         }

         // 加载玩家图片
        var playeImg = new Image();
         playeImg.src = 'img/player.png';
         playeImg.onload = function() {
         }
         play = new Player(240, 800, playerW, playerH, playeImg);

         // 加载子弹图片
         var bulletImg = new Image();
         bulletImg.src = 'img/bullet.png';
         bulletImg.onload = function() {

         function creatBullet(){

        if(play.num>0){
           for(var i=-1;i<2;i++){
               bullets.push(new Bullet(play.x -5, play.y - play.h, 32,
32,5*i,speedY, bulletImg));
           }
        }else{
               bullets.push(new Bullet(play.x -5, play.y - play.h, 32,
32,0,speedY, bulletImg));
        }





        }
        if(!play.die)
        creat_bullet=setInterval(creatBullet, 200);
```

```
       else
       clearInterval(creat_bullet);
        }
        /*
        $(window).keyup(function (evt) { // onkeyup event handle
            var pk = pressedKeys[evt.keyCode];
            if (pk) {
                delete pressedKeys[evt.keyCode]; // remove pressed key
from array
            }
            if (evt.keyCode == 65) { // 'A' button - add a rocket
              play.die=true;
              clearInterval(creat_bullet);
            }

        });

        */

        $("span").text('num:'+bullets.length);

        //玩家飞机跟随鼠标移动
        canvas.mousemove(function(e){
       // $("span").text('X:'+e.pageX + ", Y:" + e.pageY);
        play.x=e.pageX-offLeft;
        play.y=e.pageY;
        });



        setInterval(drawScene, 40); // loop drawScene
    });
```

3.1.9 爆炸清屏

Html 09 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<title>html5 飞机大战</title>
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
<link href="css/index.css" rel="stylesheet" />
<script src="js/jquery.js"></script>
<script src="js/09.js"></script>
<!--[if lt IE 9]>
<script src="js/html5.js"></script>
<![endif]-->
<style>
span{position:absolute;top:300px;right:200px;display:block;height:100px;width:200px;}
</style>
</head>
<body>
<h1><a href="./index.html">返回</a></h1>
<div id="canvas">
<canvas id="gameCanvas" width='480' height='852'>你的浏览器不支持 html5，请使用谷歌、
火狐、IE9 或更高级的浏览器</canvas>
</div>
<span></span>
</body>
</html>
```

JavaScript 09 代码如下：

```
/**
 *画布
 */
var canvas;
/**
 *画笔
 */
var paint;
```

```
/**
 *背景图移动速度
 */
var bgShiftY=0;
/**
 *背景图
 */
var bgImg;
/**
 *玩家
 */
var play;
/**
 *玩家飞机宽
 */
var playerW = 105;
/**
 *玩家飞机高
 */
var playerH = 128;
/**
 *飞机的当前桢
 */
var playFrame = 0;
var iSprDir = 4; // initial dragon direction
/**
 *子弹数组
 */
var bullets = [];
/**
 *子弹速度
 */
var speedY = 50;
var pressedKeys = []; // array of pressed keys
/**
 *e0 宽，小型飞机
 */
var iEnemyW =48; // enemy width
/**
 *e0 高，小型飞机
 */
var iEnemyH = 37; // enemy height
/**
 *e0 数组
 */
var enemies = [];

var enTimer = null; // random timer for a new enemy

var e1Timer = null;

var iEnemySpeed = 5; // initial enemy speed

var   bossTimer=null;

var e0Frame=1;

var curFrame=0;
var curFramee=0;
var bgSound; // bg sound

var iScore=0;

var iScore = 0; // total score
var iLife =50; // total life of play

var die =false; // game pause
```

```
    var press=false;

    var all_die=false;

    var clear_num=0;
    /**
     *爆炸数组
     */
    var explosions = []; // array of explosions
    /**
     *获取当前 html 元素的 x 坐标值
     */
    function pageX(elem){
      return
elem.offsetParent?(elem.offsetLeft+pageX(elem.offsetParent)):elem.off
setLeft;
    }

    /**
     *获取当前 html 元素的 y 坐标值
     */
    function pageY(elem){
      return
elem.offsetParent?(elem.offsetTop+pageY(elem.offsetParent)):elem.offs
etTop;
    }

    /**
     *清除画布
     */
    function clear() {
        paint.clearRect(0,           0,           paint.canvas.width,
paint.canvas.height);
    }
```

```
/**
 *Player 对象
 */
function Player(x, y, w, h, image,img_e,b_num) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.num=b_num;
    this.image = image;
    this.e=img_e;
    this.die = false;
}
/**
 *子弹  对象
 */
function Bullet(x, y, w, h,speedx, speedy,power, image) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.power=power;
    this.speedx = speedx;
    this.speedy = speedy;
    this.image = image;
}
/**
 *敌人  对象
 */
function          Enemy(x,          y,          w,          h,
speed,hp,image,changeFrame,totleFrame,count,score) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.hp = hp;
    this.cf = changeFrame;//要显示的桢
    this.tf = totleFrame;//总的桢数
   this.count = count;//计算用
    this.speed = speed;
    this.image = image;
    this.score = score;
}
/**
 *爆炸  对象
 */
function Explosion(x, y, w, h, sprite, image,frame) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
    this.sprite = sprite;
    this.image = image;
   this.f = frame;//飞机爆炸的桢数
}
// get random number between X and Y
function getRand(x, y) {
    return Math.floor(Math.random()*y)+x;
}




/**
 *画场景
 */
function drawScene() {
        if(!die){



        clear();
        //背景图片滚动 start
          paint.drawImage(bgImg,0,bgShiftY);
          paint.drawImage(bgImg,0,bgShiftY-852);
          bgShiftY +=4;
         if (bgShiftY >=852) bgShiftY =0;
        // end

         //玩家飞机切帧 start
         playFrame++;
         if (playFrame >=2)playFrame = 0;
         paint.drawImage(play.image,    playFrame*play.w,0,    play.w,
play.h, play.x - play.w/2, play.y - play.h/2, play.w, play.h);
        // end

             // draw bullets
             if (bullets.length > 0) {
                 for (var key in bullets) {
                     if (bullets[key] != undefined) {

                         paint.drawImage(bullets[key].image,
bullets[key].x, bullets[key].y);
                         bullets[key].y -= bullets[key].speedy;
                      bullets[key].x += bullets[key].speedx;
                         // if a rocket is out of screen - remove it
                         if (bullets[key].y < 0) {
                             delete bullets[key];
                         }
                     }
                 }
             }
```

```
            // draw explosions
            if (explosions.length > 0) {
                for (var key in explosions) {
                    if (explosions[key] != undefined) {
                        // display explosion sprites

                        paint.drawImage(explosions[key].image,
explosions[key].sprite*explosions[key].w,    0,    explosions[key].w,
explosions[key].h,    explosions[key].x    -    explosions[key].w/2,
explosions[key].y    -    explosions[key].h/2,    explosions[key].w,
explosions[key].h);
                        explosions[key].sprite++;

                        // remove an explosion object when it expires
                        if (explosions[key].sprite >explosions[key].f)
{
                            delete explosions[key];
                        }
                    }
                }
            }

            // draw enemies
            if (enemies.length > 0) {
                for (var ekey in enemies) {
                    if (enemies[ekey] != undefined) {
                     enemies[ekey].y -= enemies[ekey].speed;

       paint.drawImage(enemies[ekey].image,enemies[ekey].count*(enemi
es[ekey].w),0,enemies[ekey].w,enemies[ekey].h,enemies[ekey].x,enemies
[ekey].y,enemies[ekey].w,enemies[ekey].h);
                        enemies[ekey].count++;
                     if(enemies[ekey].count>=
enemies[ekey].cf)enemies[ekey].count=0;
                        // remove an enemy object if it is out of screen
                        if (enemies[ekey].y > canvas.height) {
                            delete enemies[ekey];
                        }
                    }
                }
            }

            if (enemies.length > 0) {
       //使用了必杀
       if(all_die){
        for (var ekey in enemies) {
                        enemies[ekey].speed=0;
       explosions.push(new Explosion(enemies[ekey].x + enemies[ekey].w
/      2,      enemies[ekey].y      +      enemies[ekey].h      /      2,
enemies[ekey].w,enemies[ekey].h,                                       0,
enemies[ekey].image,enemies[ekey].tf));
                         iScore+=enemies[ekey].score;
                         delete enemies[ekey];
       }
       all_die=false;
           }
                for (var ekey in enemies) {

                    if (enemies[ekey] != undefined) {
       //必杀
       if(all_die){
                        enemies[ekey].speed=0;
       explosions.push(new Explosion(enemies[ekey].x + enemies[ekey].w
/      2,      enemies[ekey].y      +      enemies[ekey].h      /      2,
enemies[ekey].w,enemies[ekey].h,                                       0,
enemies[ekey].image,enemies[ekey].tf));
                         iScore+=enemies[ekey].score;
```

```
                          delete enemies[ekey];

        }

                         // collisions with bullets
                         if (bullets.length > 0) {
                             for (var key in bullets) {
                                 if          (bullets[key]          !=
undefined&&enemies[ekey] != undefined) {

if(Math.pow((bullets[key].y-(enemies[ekey].y+enemies[ekey].h/2) ),2)+
Math.pow((bullets[key].x-(enemies[ekey].x+enemies[ekey].w/2)),2)<Math
.pow(enemies[ekey].w/2,2)){
        enemies[ekey].hp-=bullets[key].power;

        curFrame++;
        if(curFrame>enemies[ekey].cf)curFrame=0;
        paint.drawImage(enemies[ekey].image,
curFrame*enemies[ekey].w,0,enemies[ekey].w,enemies[ekey].h,enemies[ek
ey].x,enemies[ekey].y,enemies[ekey].w,enemies[ekey].h);

        if(enemies[ekey].hp<=0){
            enemies[ekey].speed=0;
        explosions.push(new Explosion(enemies[ekey].x + enemies[ekey].w
/      2,      enemies[ekey].y      +      enemies[ekey].h      /      2,
enemies[ekey].w,enemies[ekey].h,                                      0,
enemies[ekey].image,enemies[ekey].tf));
                                        iScore+=enemies[ekey].score;
                                        delete enemies[ekey];

        }
        delete bullets[key];

                                    }
```

```
                                }
                            }
                        }

                // collisions with play
            if (enemies[ekey] != undefined) {
                if
(Math.pow((play.y-(enemies[ekey].y+enemies[ekey].h)),2)+Math.pow((pla
y.x-(enemies[ekey].x+enemies[ekey].w/2)),2)<Math.pow(play.w/2,2)) {
                 // canvas.unbind('mousemove');
                    // delete enemy and make damage
                  //  delete play;
                   // play = new Player(240, 800, playerW, playerH,
playeImg,peImg);
         iLife -= 1;
    //
                      if (iLife <= 0) { // Game over
    die = true;//
                          // draw score
                          canvas.unbind('mousemove');
                          paint.font = '14px Verdana';
                          paint.fillStyle = '#000';
                          paint.fillText('Game  over,  your  score:  '  +
iScore + ' points', 25, 200);
                          return;
                      }
                      delete play;
                      explosions.push(new  Explosion(play.x  ,  play.y  ,
play.w, play.h, 0,play.e,10));
                }
            }
                      }
                }
            }
            paint.font = '14px Verdana';
             paint.fillStyle = '#000';
             paint.fillText('Life: ' + iLife , 50, 660);
             paint.fillText('Score: ' + iScore, 50, 50);
}
}
$(window).load(function() {
   paint=$('#gameCanvas')[0].getContext('2d');
   canvas=$('#gameCanvas');
   //画布宽高
   var width = canvas.width;
    var height = canvas.height;
   // getContext
    //画布距离浏览器左边的距离
    var offLeft=pageX(canvas[0]);
    // 加载背景图片
    bgImg = new Image();
    bgImg.src = 'img/bg.png';
    bgImg.onload = function() {
    }
    bgImg.onerror = function() {
        console.log('加载背景图片出错！');
    }
    // 加载玩家图片
   var playeImg = new Image();
    playeImg.src = 'img/player.png';
    playeImg.onload = function() {
    }
    // 加载玩家爆炸图片
   var peImg = new Image();
    peImg.src = 'img/p_e.png';
    peImg.onload = function() {
    }
    play = new Player(240, 800, playerW, playerH, playeImg,peImg,1);
```

```
        // 加载子弹图片
        var bulletImg = new Image();
        bulletImg.src = 'img/bullet.png';
        bulletImg.onload = function() {
        function creatBullet(){
           if(play.num>0){
           for(var i=-1;i<2;i++){
               bullets.push(new Bullet(play.x -5, play.y - play.h, 32,
32,5*i,speedY,20, bulletImg));
           }
       }else{
               bullets.push(new Bullet(play.x -5, play.y - play.h, 32,
32,0,speedY,20,bulletImg));
       }
       }
       if(!play.die)
       creat_bullet=setInterval(creatBullet,200);
       else
       clearInterval(creat_bullet);
        }
        // initialization of empty enemy
        var e0Img = new Image();
        e0Img.src = 'img/e0.png';
        e0Img.onload = function() {
           function addEnemy() {
                clearInterval(enTimer);
               iEnemySpeed=getRand(4,10);
              //48-432 之间
              //                    var                    randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
                var randX =Math.floor(Math.random()*432);
                enemies.push(new    Enemy(randX,    -iEnemyH,    iEnemyW,
iEnemyH, - iEnemySpeed,20,e0Img,0,4,0,1000));
           //  $("span").text('X:'+randX);
```

```
               var interval = getRand(200, 400);
               enTimer = setInterval(addEnemy, interval); // loop
           }
       addEnemy();
        }
        var e1Img = new Image();
        e1Img.src = 'img/e1.png';
        e1Img.onload = function() {
           function addEnemy() {
               clearInterval(e1Timer);
              iEnemySpeed=getRand(5,5);
             //48-432 之间
             //                       var                       randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
               var randX =Math.floor(Math.random()*432);
               enemies.push(new   Enemy(randX,   -68,   68,   94,   -
iEnemySpeed,60,e1Img,1,5,0,5000));
          //  $("span").text('X:'+randX);
               var interval = getRand(1000, 4000);
               e1Timer = setInterval(addEnemy, interval); // loop
           }
       addEnemy();
        }

        var bossImg = new Image();
        bossImg.src = 'img/boss2.png';
        bossImg.onload = function() {

           function addEnemy() {
               clearInterval(bossTimer);
              iEnemySpeed=getRand(5,5);
             //48-432 之间
             //                       var                       randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
```

```
                var randX =Math.floor(Math.random()*332);
                enemies.push(new    Enemy(randX,    -257,172,257,    -
iEnemySpeed,200,bossImg,2,10,0,30000));
            //  $("span").text('X:'+randX);
                var interval = getRand(4000, 8000);
                bossTimer = setInterval(addEnemy, interval); // loop
            }
        addEnemy();
         }
         $(window).keyup(function (evt) { // onkeyup event handle
              var pk = pressedKeys[evt.keyCode];
              if (pk) {
                  delete pressedKeys[evt.keyCode]; // remove pressed key
from array
              }
              if (evt.keyCode == 65) { // 'A' button - add a rocket
               all_die=true;
              }
         });
        // $("span").text('num:'+bullets.length);
              // 'bg' music init

         // bgSound = new Audio('media/wj.wav');
         // bgSound.volume = 0.9;
         // bgSound.addEventListener('ended', function() { // looping bg
sound
              // this.currentTime = 0;
              // this.play();
         // }, false);
         // bgSound.play();
         //玩家飞机跟随鼠标移动
           canvas.mousemove(function(e){
        // $("span").text('X:'+e.pageX + ", Y:" + e.pageY);
         play.x=e.pageX-offLeft;
         play.y=e.pageY;
         if(!press)return;
         });
         setInterval(drawScene,30); // loop drawScene

     });
```

3.1.10 奖励事件

Html 10 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<title>html5 飞机大战</title>
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
<link href="css/index.css" rel="stylesheet" />
<script src="js/jquery.js"></script>
<script src="js/10.js"></script>
<!--[if lt IE 9]>
<script src="js/html5.js"></script>
<![endif]-->
<style>
span{position:absolute;top:300px;right:200px;display:block;height:100px;width:200px;}
</style>
</head>
<body>
<h1><a href="./index.html">返回</a></h1>
<div id="canvas">
<canvas id="gameCanvas" width='480' height='852'>你的浏览器不支持 html5，请使用谷歌、
火狐、IE9 或更高级的浏览器</canvas>
</div>
<span></span>
</body>
</html>
```

JavaScript 10 代码如下：

```
/**
 *画布
 */
var canvas;
/**
 *画笔
 */
var paint;
/**
 *背景图移动速度
 */
var bgShiftY=0;
/**
 *背景图
 */
var bgImg;
/**
 *玩家
 */
var play;
/**
 *玩家飞机宽
 */
var playerW = 105;
/**
 *玩家飞机高
 */
var playerH = 128;
/**
 *飞机的当前桢
 */
var playFrame = 0;
var iSprDir = 4; // initial dragon direction
```

```
/**
 *子弹数组
 */
var bullets = [];
/**
 *子弹速度
 */
var speedY = 50;

var pressedKeys = []; // array of pressed keys
/**
 *e0 宽，小型飞机
 */
var iEnemyW =48; // enemy width
/**
 *e0 高，小型飞机
 */
var iEnemyH = 37; // enemy height
/**
 *e0 数组
 */
var enemies = [];
var bonusarr=[];
var enTimer = null; // random timer for a new enemy
var e1Timer = null;
var iEnemySpeed = 5; // initial enemy speed
var bossTimer=null;
var bonusTimer=null;
var e0Frame=1;
var curFrame=0;
var curFramee=0;
var bgSound; // bg sound

var iScore = 0; // total score
```

```
    var iLife =50; // total life of play

    var die =false; // game pause
    var press=false;

    var all_die=false;
    /**
     *上升 false
     */
    var up=false;
    var clear_num=0;
    /**
     *爆炸数组
     */
    var explosions = []; // array of explosions
    /**
     *获取当前 html 元素的 x 坐标值
     */
    function pageX(elem){
      return
elem.offsetParent?(elem.offsetLeft+pageX(elem.offsetParent)):elem.off
setLeft;
    }

    /**
     *获取当前 html 元素的 y 坐标值
     */
    function pageY(elem){
      return
elem.offsetParent?(elem.offsetTop+pageY(elem.offsetParent)):elem.offs
etTop;
    }

    /**
```

```
     *清除画布
     */
    function clear() {
        paint.clearRect(0,           0,           paint.canvas.width,
paint.canvas.height);
    }
    /**
     *Bonus 对象
     */
    function Bonus(x, y, w, h,speedx, speedy,image,curFrame,type) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.speedx = speedx;
        this.speedy = speedy;
        this.f=curFrame;
        this.type=type;
        this.image = image;
    }
    /**
     *Player 对象
     */
    function Player(x, y, w, h, image,img_e,b_num,bonus) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.num=b_num;//多发子弹
        this.b=bonus;//必杀
        this.image = image;
        this.e=img_e;
        this.die = false;
    }
    /**
     *子弹  对象
     */
    function Bullet(x, y, w, h,speedx, speedy,power, image) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.power=power;
        this.speedx = speedx;
        this.speedy = speedy;
        this.image = image;
    }
    /**
     *敌人  对象
     */
    function          Enemy(x,          y,          w,          h,
speed,hp,image,changeFrame,totleFrame,count,score) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.hp = hp;
        this.cf = changeFrame;//要显示的桢
        this.tf = totleFrame;//总的桢数
       this.count = count;//计算用
        this.speed = speed;
        this.image = image;
        this.score = score;
    }
    /**
     *爆炸  对象
     */
    function Explosion(x, y, w, h, sprite, image,frame) {
        this.x = x;
        this.y = y;
        this.w = w;
        this.h = h;
        this.sprite = sprite;
        this.image = image;
       this.f = frame;//飞机爆炸的桢数
    }
    // get random number between X and Y
    function getRand(x, y) {
        return Math.floor(Math.random()*y)+x;
    }
    /**
     *画场景
     */
    function drawScene() {
       if(!die){
       clear();
       //背景图片滚动 start
         paint.drawImage(bgImg,0,bgShiftY);
         paint.drawImage(bgImg,0,bgShiftY-852);
         bgShiftY +=4;
        if (bgShiftY >=852) bgShiftY =0;
       // end
        //玩家飞机切帧 start
        playFrame++;
        if (playFrame >1)playFrame = 0;
        paint.drawImage(play.image,    playFrame*play.w,0,    play.w,
play.h, play.x - play.w/2, play.y - play.h/2, play.w, play.h);
       // end
            // draw bonus
            if (bonusarr.length > 0) {
                for (var key in bonusarr) {
                    if (bonusarr[key] != undefined) {
                         //        paint.drawImage(bonusarr[key].image,
bonusarr[key].x, bonusarr[key].y);
                      paint.drawImage(bonusarr[key].image,
bonusarr[key].w*bonusarr[key].f,0,bonusarr[key].w,bonusarr[key].h,bon
usarr[key].x - bonusarr[key].w/2, bonusarr[key].y - bonusarr[key].h/2,
bonusarr[key].w, bonusarr[key].h);
                          bonusarr[key].y += bonusarr[key].speedy;
                         if(bonusarr[key].y>500){   //大于 200
       if(!up){
           bonusarr[key].y -=bonusarr[key].speedy;
           bonusarr[key].speedy-=10;
           if(bonusarr[key].speedy<100){
           //  bonusarr[key].y += 20;
               up=true;
           }
       }else{
           bonusarr[key].y += 20;
           up=false;
       }
                                }
                         if (bonusarr[key].y< -canvas.height/2) {
                             delete bonusarr[key];
                         }
                     }
                 }
                    for (var ekey in bonusarr) {
                     if (bonusarr[ekey] != undefined) {
       if
(Math.pow((play.y-(bonusarr[ekey].y+bonusarr[ekey].h)),2)+Math.pow(((
play.x+play.w/2)-(bonusarr[ekey].x+bonusarr[ekey].w/2)),2)<Math.pow(p
lay.h/2,2)) {
                   if(bonusarr[ekey].type==0){
                      play.b++;
                   }else if(bonusarr[ekey].type==1){
                      play.num=2;
                    }

                     delete bonusarr[key];
               }
                     }

                    }
             }
             // draw bullets
             if (bullets.length > 0) {
                 for (var key in bullets) {
                     if (bullets[key] != undefined) {

                         paint.drawImage(bullets[key].image,
bullets[key].x, bullets[key].y);
                         bullets[key].y -= bullets[key].speedy;
                      bullets[key].x += bullets[key].speedx;
                         // if a rocket is out of screen - remove it
                         if (bullets[key].y < 0) {
                             delete bullets[key];
                         }
                     }
                 }
             }
             // draw explosions
             if (explosions.length > 0) {
                 for (var key in explosions) {
                     if (explosions[key] != undefined) {
                         // display explosion sprites
                         paint.drawImage(explosions[key].image,
explosions[key].sprite*explosions[key].w,     0,     explosions[key].w,
explosions[key].h,     explosions[key].x     -     explosions[key].w/2,
explosions[key].y     -     explosions[key].h/2,     explosions[key].w,
explosions[key].h);
                         explosions[key].sprite++;
                         // remove an explosion object when it expires
                         if (explosions[key].sprite >explosions[key].f)
{
                             delete explosions[key];
                         }
                     }
                 }
             }

             // draw enemies
             if (enemies.length > 0) {
                 for (var ekey in enemies) {
                     if (enemies[ekey] != undefined) {
                      enemies[ekey].y -= enemies[ekey].speed;
        paint.drawImage(enemies[ekey].image,enemies[ekey].count*(enemi
es[ekey].w),0,enemies[ekey].w,enemies[ekey].h,enemies[ekey].x,enemies
[ekey].y,enemies[ekey].w,enemies[ekey].h);
                         enemies[ekey].count++;
                      if(enemies[ekey].count>=
enemies[ekey].cf)enemies[ekey].count=0;
                         // remove an enemy object if it is out of screen
                         if (enemies[ekey].y > canvas.height) {
                             delete enemies[ekey];
                         }
                     }
                 }
             }

             if (enemies.length > 0) {
        //使用了必杀
        if(all_die){
         for (var ekey in enemies) {
```

```
                    enemies[ekey].speed=0;
        explosions.push(new Explosion(enemies[ekey].x + enemies[ekey].w
/      2,      enemies[ekey].y      +      enemies[ekey].h      /      2,
enemies[ekey].w,enemies[ekey].h,                                       0,
enemies[ekey].image,enemies[ekey].tf));
                  //   iScore+=parseFloat(enemies[ekey].score);
                 //   iScore=enemies[ekey].score+iScore;
                    delete enemies[ekey];
        }
        all_die=false;
            }
               for (var ekey in enemies) {

                   if (enemies[ekey] != undefined) {

                       // collisions with bullets
                       if (bullets.length > 0) {
                           for (var key in bullets) {
                               if          (bullets[key]          !=
undefined&&enemies[ekey] != undefined) {

if(Math.pow((bullets[key].y-(enemies[ekey].y+enemies[ekey].h/2) ),2)+
Math.pow((bullets[key].x-(enemies[ekey].x+enemies[ekey].w/2)),2)<Math
.pow(enemies[ekey].w/2+10,2)){
        enemies[ekey].hp-=bullets[key].power;

        curFrame++;
        if(curFrame>enemies[ekey].cf)curFrame=0;
        paint.drawImage(enemies[ekey].image,
curFrame*enemies[ekey].w,0,enemies[ekey].w,enemies[ekey].h,enemies[ek
ey].x,enemies[ekey].y,enemies[ekey].w,enemies[ekey].h);

        if(enemies[ekey].hp<=0){
            enemies[ekey].speed=0;
```

```
       explosions.push(new Explosion(enemies[ekey].x + enemies[ekey].w
/      2,      enemies[ekey].y      +      enemies[ekey].h      /      2,
enemies[ekey].w,enemies[ekey].h,                                       0,
enemies[ekey].image,enemies[ekey].tf));
                                       iScore+=enemies[ekey].score;
                                       delete enemies[ekey];

       }
       delete bullets[key];

                                    }
                                }
                            }
                        }
                // collisions with play
            if (enemies[ekey] != undefined) {
                if
(Math.pow((play.y-(enemies[ekey].y+enemies[ekey].h)),2)+Math.pow((pla
y.x-(enemies[ekey].x+enemies[ekey].w/2)),2)<Math.pow(play.w/2,2)) {
                    play.num=0;
                     iLife -= 1;
    //
                     if (iLife <= 0) { // Game over
    die = true;//
                        // draw score
                        canvas.unbind('mousemove');
                        paint.font = '14px Verdana';
                        paint.fillStyle = '#000';
                        paint.fillText('Game  over,  your  score:  '  +
iScore + ' points', 25, 200);
                        return;
                    }

                    delete play;
                    explosions.push(new  Explosion(play.x  ,  play.y  ,
play.w, play.h, 0,play.e,10));


                }
            }
                    }
                }
            }
            paint.font = '14px Verdana';
            paint.fillStyle = '#000';
            paint.fillText('Life: ' + iLife , 5, 660);
            // paint.fillText('必杀: ' + play.b , 50, 680);
            paint.fillText('Score: ' + iScore, 50, 50);

            if(play.b>0){


              if(play.b>4){

paint.drawImage(play.image,227,38,69,60,0,680,69,60);
                   paint.fillText('X ' + play.b , 70, 720);
              }else{
                  for(var i=0;i<play.b;i++){
                  paint.drawImage(play.image,227,38,69,60,70*i,
680,69,60);

                  }




                     }
                   }
    }
    }
    $(window).load(function() {
       paint=$('#gameCanvas')[0].getContext('2d');
       canvas=$('#gameCanvas');
       //画布宽高
       var width = canvas.width;
        var height = canvas.height;

       // getContext
        //画布距离浏览器左边的距离
        var offLeft=pageX(canvas[0]);
        // 加载背景图片
        bgImg = new Image();
        bgImg.src = 'img/bg.png';
        bgImg.onload = function() {
        }
        bgImg.onerror = function() {
            console.log('加载背景图片出错！');
        }
        // 加载玩家图片
       var playeImg = new Image();
        playeImg.src = 'img/player1.png';
        playeImg.onload = function() {
        }
        // 加载玩家爆炸图片
       var peImg = new Image();
        peImg.src = 'img/p_e.png';
        peImg.onload = function() {
        }
        play    =    new    Player(240,    800,    playerW,    playerH,
playeImg,peImg,0,0);



        // 加载子弹图片
        var bulletImg = new Image();
        bulletImg.src = 'img/bullet.png';
    bulletImg.onload = function() {
        function creatBullet(){
           if(play.num>0){
           for(var i=-1;i<2;i++){
               bullets.push(new Bullet(play.x -5, play.y - play.h, 32,
32,5*i,speedY,20, bulletImg));
           }
       }else{
               bullets.push(new Bullet(play.x -5, play.y - play.h, 32,
32,0,speedY,20,bulletImg));
       }
       }
       if(!play.die)
       creat_bullet=setInterval(creatBullet,200);
       else
       clearInterval(creat_bullet);
        }
        // initialization of empty enemy
        var e0Img = new Image();
        e0Img.src = 'img/e0.png';
        e0Img.onload = function() {

           function addEnemy() {
               clearInterval(enTimer);
               iEnemySpeed=getRand(4,10);
              //48-432 之间
              //                      var                        randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
               var randX =Math.floor(Math.random()*432);
               enemies.push(new   Enemy(randX,   -iEnemyH,   iEnemyW,
iEnemyH, - iEnemySpeed,20,e0Img,0,4,0,1000));
           // $("span").text('X:'+randX);
```

```
            var interval = getRand(100, 400);
            enTimer = setInterval(addEnemy, interval); // loop
        }
    addEnemy();
     }



     var e1Img = new Image();
     e1Img.src = 'img/e1.png';
     e1Img.onload = function() {

        function addEnemy() {
            clearInterval(e1Timer);
           iEnemySpeed=getRand(5,5);
           //48-432 之间
           //                        var                        randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
            var randX =Math.floor(Math.random()*432);
            enemies.push(new   Enemy(randX,   -68,   68,   94,   -
iEnemySpeed,60,e1Img,1,5,0,5000));
        // $("span").text('X:'+randX);
            var interval = getRand(1000, 4000);
            e1Timer = setInterval(addEnemy, interval); // loop
        }
    addEnemy();
     }

     var bossImg = new Image();
     bossImg.src = 'img/boss2.png';
     bossImg.onload = function() {

        function addEnemy() {
            clearInterval(bossTimer);
           iEnemySpeed=getRand(5,5);
```

```
              //48-432 之间
              //                       var                       randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
               var randX =Math.floor(Math.random()*332)+20;
               enemies.push(new    Enemy(randX,    -257,172,257,     -
iEnemySpeed,200,bossImg,2,10,0,30000));
           //  $("span").text('X:'+randX);
               var interval = getRand(4000, 8000);
               bossTimer = setInterval(addEnemy, interval); // loop
           }
       addEnemy();
        }
        var bonusImg = new Image();
        bonusImg.src = 'img/Bonus.png';
        bonusImg.onload = function() {

           function addEnemy() {
               clearInterval(bonusTimer);
              iEnemySpeed=getRand(5,5);
              //48-432 之间
              //                       var                       randX
=Math.floor(Math.random()*(480-iEnemyW))+iEnemyW;
               var randX =Math.floor(Math.random()*432);
               var ran =Math.floor(Math.random()*2);
               bonusarr.push(new                          Bonus(randX,
-257,88,111,0,iEnemySpeed,bonusImg,ran,ran));

           //  $("span").text('必杀:'+play.b);
               var interval = getRand(8000, 11000);
               bonusTimer = setInterval(addEnemy, interval); // loop
           }
       addEnemy();
        }
        $(window).keyup(function (evt) { // onkeyup event handle
            var pk = pressedKeys[evt.keyCode];
            if (pk) {
                delete pressedKeys[evt.keyCode]; // remove pressed key
from array
            }
            if (evt.keyCode == 65) { // 'A' button - add a rocket
            if(play.b>0){
       all_die=true;
       play.b--;
            }



            }
        });
       // $("span").text('num:'+bullets.length);
            // 'bg' music init
        // bgSound = new Audio('media/wj.wav');
        // bgSound.volume = 0.9;
        // bgSound.addEventListener('ended', function() { // looping bg
sound
            // this.currentTime = 0;
            // this.play();
        // }, false);
        // bgSound.play();
        //玩家飞机跟随鼠标移动
          canvas.mousemove(function(e){
       // $("span").text('X:'+e.pageX + ", Y:" + e.pageY);
        play.x=e.pageX-offLeft;
        play.y=e.pageY;
        if(!press)return;
        });
        setInterval(drawScene,30); // loop drawScene
    });
```

3.1.11 最终项目展示

HTML 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
<title>html5 飛機大戰</title>
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
<link href="css/index.css" rel="stylesheet" />
<script src="js/jquery.js"></script>
<script src="js/1.js"></script>
<!--[if lt IE 9]>
<script src="js/html5.js"></script>
<![endif]-->
<style>
span{position:absolute;top:300px;right:200px;display:block;height:100px;width:200px;}
</style>
</head>
<body>
<div id="message">
<h1><a href="./index.html">返回</a></h1>
<ol>
<li>用鼠标控制飞机移动</li>
<li>按键盘上的 F11 可以切换全屏</li>
<li>三发子弹的奖励时间是 20 秒</li>
<li>按键盘上的 A 可以使用必杀</li>
</ol>
<h2>tiandian 制作</h2>
<h2>email:82944930@qq.com</h2>
</div>

<div id="canvas">
<canvas id="gameCanvas" width='480' height='852'>你的浏览器不支持 html5，请使用谷歌、
火狐、IE9 或更高级的浏览器</canvas>
</div>
<span></span>
```

```
</body>
</html>
```

3.2 HTML 5 多样数据 RGraph 插件制作饼图项目

我们都知道统计图是利用点、线、面、体等绘制成几何图形，以表示各种数量间的关系及其变动情况的工具。表现统计数字大小和变动的各种图形总称。其中有条形统计图、扇形统计图、折线统计图、象形图等。在统计学中把利用统计图形表现统计资料的方法叫做统计图示法。其特点是：形象具体、简明生动、通俗易懂、一目了然。其主要用途有：表示现象间的对比关系；揭露总体结构；检查计划的执行情况；揭示现象间的依存关系，反映总体单位的分配情况；说明现象在空间上的分布情况。一般采用直角坐标系．横坐标用来表示事物的组别或自变量 x，纵坐标常用来表示事物出现的次数或因变量 y；或采用角度坐标(如圆形图)、地理坐标(如地形图)等。按图尺的数字性质分类，有实数图、累积数图、百分数图、对数图、指数图等；其结构包括图名、图目(图中的标题)、图尺(坐标单位)、各种图线(基线、轮廓线、指导线等)、图注(图例说明、资料来源等)等。

那么我们第 2 个案例项目就是来绘制各种版本的统计图。

3.2.1 HTML 页面的代码

首先我们先看一下 HTML 页面的代码：

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8">
<title>使用 RGraph 插件制作饼图</title>
<script src="RGraph.common.core.js"></script>
<script src="RGraph.pie.js"></script>
<script src="RGraph.common.tooltips.js"></script>
<script>
function window_onload()
{
//绘制饼图，获取饼图数据
var pie=new
 RGraph.Pie('myCanvas',[12000,13000,14000,15000,30000,19000]);
```

```
    //绘制饼图标题
    pie.Set('chart.title', '2010 年常州第一百货公司彩电销售分布图');
    //绘制饼图标签文字
    pie.Set('chart.labels', ['长虹（12%）', '康佳（13%）', '创维（14%）
', '三星（15%）',
    '夏普（29%）','索尼（17%）']);
    //指定饼图分隔线宽
    pie.Set('chart.linewidth', 5);
    //指定饼图分隔线颜色
    pie.Set('chart.strokestyle','white');
    //指定工具条提示信息的出现效果为淡入效果
    pie.Set('chart.tooltips.effect', 'fade');
    //指定当鼠标指针在饼块上移动时出现工具条提示信息
    pie.Set('chart.tooltips.event', 'onmousemove');
    //指定工具条提示信息的文字
    pie.Set('chart.tooltips', ['长虹（12%）','康佳（13%）','创维（14%）
','三星（15%）',
    '夏普（29%)','索尼（17%）']);
    //指定工具条提示信息具有 3d 效果
    pie.Set('chart.highlight.style', '3d');
    //绘制饼图
    pie.Draw();
    }
    </script>
    </head>
    <body onload="window_onload()">
    <h1>使用 RGraph 插件制作饼图</h1>
    <canvas id="myCanvas" width="700" height="400">
    [您的浏览器不支持 canvas 元素]
    </canvas>
    </body>
    </html>
```

3.2.2 RGraph.common.core.js 文件代码

```
    if (typeof(RGraph) == 'undefined') RGraph = {isRGraph:true,type:'common'};


    RGraph.Registry             = {};
    RGraph.Registry.store = [];
    RGraph.Registry.store['chart.event.handlers'] = [];
    RGraph.background           = {};
    RGraph.objects              = [];
    RGraph.Resizing             = {};
    RGraph.events               = [];
    /**
    * Returns five values which are used as a nice scale
    *
    * @param    max int     The maximum value of the graph
    * @param    obj object The graph object
    * @return       array   An appropriate scale
    */
    RGraph.getScale = function (max, obj)
    {
        /**
        * Special case for 0
        */
        if (max == 0) {
            return ['0.2', '0.4', '0.6', '0.8', '1.0'];
        }
        var original_max = max;
        /**
        * Manually do decimals
        */
        if (max <= 1) {
            if (max > 0.5) {
                return [0.2,0.4,0.6,0.8, Number(1).toFixed(1)];

            } else if (max >= 0.1) {
return obj.Get('chart.scale.round') ? [0.2,0.4,0.6,0.8,1] : [0.1,0.2,0.3,0.4,0.5];
            } else {
                var tmp = max;
                var exp = 0;
```

```
                    while (tmp < 1.01) {
                        exp += 1;
                        tmp *= 10;
                    }
                    var ret = ['2e-' + exp, '4e-' + exp, '6e-' + exp, '8e-' + exp, '10e-' + exp];

                    if (max <= ('5e-' + exp)) {
                        ret = ['1e-' + exp, '2e-' + exp, '3e-' + exp, '4e-' + exp, '5e-' + exp];
                    }
                    return ret;
                }
            }
            // Take off any decimals
if (String(max).indexOf('.') > 0) {
                max = String(max).replace(/\.\d+$/, '');
            }
            var interval = Math.pow(10, Number(String(Number(max)).length - 1));
           var topValue = interval;
            while (topValue < max) {
                topValue += (interval / 2);
            }
            // Handles cases where the max is (for example) 50.5
            if (Number(original_max) > Number(topValue)) {
                topValue += (interval / 2);
            }
            // Custom if the max is greater than 5 and less than 10
            if (max < 10) {
                topValue = (Number(original_max) <= 5 ? 5 : 10);
            }
            /**
            * Added 02/11/2010 to create "nicer" scales
            */
            if (obj && typeof(obj.Get('chart.scale.round')) == 'boolean' &&
obj.Get('chart.scale.round')) {
                topValue = 10 * interval;
            }
            return [topValue * 0.2, topValue * 0.4, topValue * 0.6, topValue * 0.8, topValue];
        }
        /**
        * Returns the maximum numeric value which is in an array
        *
        * @param    array arr The array
        * @param    int          Whether to ignore signs (ie negative/positive)
        * @return int             The maximum value in the array
        */
        RGraph.array_max = function (arr)
        {
                var max = null;

                for (var i=0; i<arr.length; ++i) {
                        if (typeof(arr[i]) == 'number') {

var val = arguments[1] ? Math.abs(arr[i]) : arr[i];

                                if (typeof(max) == 'number') {
                                        max = Math.max(max, val);
                                } else {
                                        max = val;
                                }
                        }
                }

                return max;
        }
        /**
        * Returns the maximum value which is in an array
        *
        * @param    array arr The array
        * @param    int      len The length to pad the array to
        * @param    mixed          The value to use to pad the array (optional)
        */
        RGraph.array_pad = function (arr, len)
        {
                if (arr.length < len) {
var val = arguments[2] ? arguments[2] : null;

                        for (var i=arr.length; i<len; ++i) {
                                arr[i] = val;
                        }
                }

                return arr;
        }
        /**
        * An array sum function
        *
        * @param    array arr The    array to calculate the total of
        * @return int              The summed total of the arrays elements
        */
        RGraph.array_sum = function (arr)
        {
                // Allow integers
                if (typeof(arr) == 'number') {
                        return arr;
                }
                var i, sum;
                var len = arr.length;
                for(i=0,sum=0;i<len;sum+=arr[i++]);
                return sum;
        }
        /**
        * A simple is_array() function
        *
        * @param    mixed obj The object you want to check
        * @return bool           Whether the object is an array or not
        */
        RGraph.is_array = function (obj)
        {
                return obj != null && obj.constructor.toString().indexOf('Array') != -1;
        }


        /**
        * Converts degrees to radians
        *
        * @param    int degrees The number of degrees
        * @return float             The number of radians
        */
        RGraph.degrees2Radians = function (degrees)
        {
                return degrees * (Math.PI / 180);
        }


        /**
        * This function draws an angled line. The angle is cosidered to be clockwise
        *
        * @param obj ctxt      The context object
        * @param int x          The X position
        * @param int y          The Y position
        * @param int angle    The angle in RADIANS
```

```
        * @param int length The length of the line
        */
        RGraph.lineByAngle = function (context, x, y, angle, length)
        {
                context.arc(x, y, length, angle, angle, false);
                context.lineTo(x, y);
                context.arc(x, y, length, angle, angle, false);
        }
        /**
        * This is a useful function which is basically a shortcut for drawing left, right, top and bottom
alligned text.
        *
        * @param object context The context
        * @param string font        The font
        * @param int        size        The size of the text
        * @param int        x              The X coordinate
        * @param int        y              The Y coordinate
        * @param string text        The text to draw
        * @parm    string                  The vertical alignment. Can be null. "center" gives center
aligned    text, "top" gives top aligned text.
        *                                      Anything else produces bottom aligned text. Default is bottom.
        * @param    string               The horizontal alignment. Can be null. "center" gives center
aligned    text, "right" gives right aligned text.
        *                                      Anything else produces left aligned text. Default is left.
        * @param    bool                   Whether to show a bounding box around the text. Defaults
not to
        * @param int                       The angle that the text should be rotate at (IN DEGREES)
        * @param string                  Background color for the text
        * @param bool                     Whether the text is bold or not
        * @param bool                     Whether the bounding box has a placement indicator
        */
        RGraph.Text = function (context, font, size, x, y, text)
        {
                /**
                * This calls the text function recursively to accommodate multi-line text
                */
                if (typeof(text) == 'string' && text.match(/\r\n/)) {

                        var arr = text.split('\r\n');

                        text = arr[0];
                        arr = RGraph.array_shift(arr);

                        var nextline = arr.join('\r\n')
RGraph.Text(context, font, size, arguments[9] == -90 ? (x + (size * 1.5)) : x, y + (size * 1.5), nextline, arguments[6] ? arguments[6] : null, 'center', arguments[8], arguments[9], arguments[10], arguments[11], arguments[12]);
                }



                // Accommodate MSIE
                if (RGraph.isIE8()) {
                        y += 2;
                }



                context.font = (arguments[11] ? 'Bold ': '') + size + 'pt ' + font;

                var i;
                var origX = x;
                var origY = y;
                var originalFillStyle = context.fillStyle;
                var originalLineWidth = context.lineWidth;

                // Need these now the angle can be specified, ie defaults for the former two args
if (typeof(arguments[6]) == null) arguments[6]   = 'bottom'; // Vertical alignment. Default to bottom/baseline
if (typeof(arguments[7]) == null) arguments[7]   = 'left';     // Horizontal alignment. Default to left
if (typeof(arguments[8]) == null) arguments[8]   = null;         // Show a bounding box. Useful for positioning during development. Defaults to false
if (typeof(arguments[9]) == null) arguments[9]   = 0;              // Angle (IN DEGREES) that the text should be drawn at. 0 is middle right, and it goes clockwise
                if (typeof(arguments[12]) == null) arguments[12] = true;       // Whether the bounding box has the placement indicator

                // The alignment is recorded here for purposes of Opera compatibility
                if (navigator.userAgent.indexOf('Opera') != -1) {
                        context.canvas.__rgraph_valign__ = arguments[6];
                        context.canvas.__rgraph_halign__ = arguments[7];
                }

                // First, translate to x/y coords
                context.save();

                        context.canvas.__rgraph_originalx__ = x;
                        context.canvas.__rgraph_originaly__ = y;
                        context.translate(x, y);
                        x = 0;
                        y = 0;

                        // Rotate the canvas if need be
                        if (arguments[9]) {
                                context.rotate(arguments[9] / 57.3);
                        }

                        // Vertical alignment - defaults to bottom
                        if (arguments[6]) {
                                var vAlign = arguments[6];

                                if (vAlign == 'center') {
                                        context.translate(0, size / 2);
                                } else if (vAlign == 'top') {
                                        context.translate(0, size);
                                }
                        }


                        // Hoeizontal alignment - defaults to left
                        if (arguments[7]) {
                                var hAlign = arguments[7];
                                var width    = context.measureText(text).width;

                                if (hAlign) {
                                        if (hAlign == 'center') {
                                                context.translate(-1 * (width / 2), 0)
                                        } else if (hAlign == 'right') {
                                                context.translate(-1 * width, 0)
                                        }
                                }
                        }


                        context.fillStyle = originalFillStyle;

                        /**
                        * Draw a bounding box if requested
                        */
                        context.save();
                                  context.fillText(text,0,0);
```

```
                                    context.lineWidth = 0.5;

                                  if (arguments[8]) {

                                          var width = context.measureText(text).width;
var ieOffset = RGraph.isIE8() ? 2 : 0;

                                          context.translate(x, y);
                                          context.strokeRect(0 - 3, 0 - 3 - size - ieOffset, width + 6, 0 + size + 6);

                                          /**
                                          * If requested, draw a background for the text
                                          */
                                          if (arguments[10]) {

                                                  var offset = 3;
var ieOffset = RGraph.isIE8() ? 2 : 0;
                                                  var width = context.measureText(text).width

                                                  //context.strokeStyle = 'gray';
                                                  context.fillStyle = arguments[10];
                                                  context.fillRect(x - offset, y - size - offset - ieOffset, width + (2 *
offset), size + (2 * offset));
                                                  //context.strokeRect(x - offset, y - size - offset - ieOffset, width + (2
* offset), size + (2 * offset));
                                          }

                                          /**
                                          * Do the actual drawing of the text
                                          */
                                          context.fillStyle = originalFillStyle;
                                          context.fillText(text,0,0);

                                          if (arguments[12]) {
                                                  context.fillRect(
arguments[7] == 'left' ? 0 : (arguments[7] == 'center' ? width / 2 : width ) - 2,
arguments[6] == 'bottom' ? 0 : (arguments[6] == 'center' ? (0 - size) / 2 : 0 - size) - 2,
                                                          4,
                                                          4
                                                  );
                                          }
                                  }
                          context.restore();
```

```
            // Reset the lineWidth
            context.lineWidth = originalLineWidth;

        context.restore();
    }



    /**
    * Clears the canvas by setting the width. You can specify a colour if you wish.
    *
    * @param object canvas The canvas to clear
    */
    RGraph.Clear = function (canvas)
    {
        var context = canvas.getContext('2d');

        /**
        * Can now clear the canvas back to fully transparent
        */
        if (!arguments[1] || (arguments[1] && arguments[1] == 'transparent')) {
            context.fillStyle = 'rgba(0,0,0,0)';
            context.globalCompositeOperation = 'source-in';
            context = canvas.getContext('2d');
            context.beginPath();
            context.fillRect(-1000,-1000,canvas.width + 2000,canvas.height + 2000);
            context.fill();

            // Reset the globalCompositeOperation
            context.globalCompositeOperation = 'source-over';

        } else {
            context.fillStyle = arguments[1];
            context = canvas.getContext('2d');
            context.beginPath();
            context.fillRect(-1000,-1000,canvas.width + 2000,canvas.height + 2000);
            context.fill();
        }

        // Don't do this as it also clears any translation that may have occurred
        //canvas.width = canvas.width;

        if (RGraph.ClearAnnotations) {
            RGraph.ClearAnnotations(canvas.id);
        }
```

```
            RGraph.FireCustomEvent(canvas.__object__, 'onclear');
        }



        /**
        * Draws the title of the graph
        *
        * @param object    canvas The canvas object
        * @param string    text       The title to write
        * @param integer gutter The size of the gutter
        * @param integer                The center X point (optional - if not given it will be generated
from the canvas width)
        * @param integer                Size of the text. If not given it will be 14
        */
        RGraph.DrawTitle = function (canvas, text, gutter)
        {
            var obj          = canvas.__object__;
            var context = canvas.getContext('2d');
var size        = arguments[4] ? arguments[4] : 12;
            var centerx = (arguments[3] ? arguments[3] : RGraph.GetWidth(obj) / 2);
            var keypos    = obj.Get('chart.key.position');
            var vpos        = gutter / 2;
            var hpos        = obj.Get('chart.title.hpos');
            var bgcolor = obj.Get('chart.title.background');

            // Account for 3D effect by faking the key position
            if (obj.type == 'bar' && obj.Get('chart.variant') == '3d') {
                keypos = 'gutter';
            }

            context.beginPath();
            context.fillStyle = obj.Get('chart.text.color') ? obj.Get('chart.text.color') : 'black';

            /**
            * Vertically center the text if the key is not present
            */
            if (keypos && keypos != 'gutter') {
                var vCenter = 'center';

            } else if (!keypos) {
                var vCenter = 'center';

            } else {
                var vCenter = 'bottom';
            }

            // if chart.title.vpos does not equal 0.5, use that
            if (typeof(obj.Get('chart.title.vpos')) == 'number') {
                vpos = obj.Get('chart.title.vpos') * gutter;
            }

            // if chart.title.hpos is a number, use that. It's multiplied with the (entire) canvas width
            if (typeof(hpos) == 'number') {
                centerx = hpos * canvas.width;
            }

            // Set the colour
            if (typeof(obj.Get('chart.title.color') != null)) {
                var oldColor = context.fillStyle
                var newColor = obj.Get('chart.title.color')
                context.fillStyle = newColor ? newColor : 'black';
            }

            /**
            * Default font is Verdana
            */
            var font = obj.Get('chart.text.font');

            /**
            * Draw the title itself
            */
            RGraph.Text(context, font, size, centerx, vpos, text, vCenter, 'center', bgcolor != null,
null, bgcolor, true);

            // Reset the fill colour
            context.fillStyle = oldColor;
        }


        /**
        * This function returns the mouse position in relation to the canvas
        *
        * @param object e The event object.
        */
        RGraph.getMouseXY = function (e)
        {
            var obj = (RGraph.isIE8() ? event.srcElement : e.target);
            var x;
            var y;

            if (RGraph.isIE8()) e = event;

            // Browser with offsetX and offsetY
            if (typeof(e.offsetX) == 'number' && typeof(e.offsetY) == 'number') {
                x = e.offsetX;
                y = e.offsetY;

            // FF and other
            } else {
                x = 0;
                y = 0;

                while (obj != document.body && obj) {
                    x += obj.offsetLeft;
                    y += obj.offsetTop;

                    obj = obj.offsetParent;
                }

                x = e.pageX - x;
                y = e.pageY - y;
            }

            return [x, y];
        }


        /**
        * This function returns a two element array of the canvas x/y position in
        * relation to the page
        *
        * @param object canvas
        */
        RGraph.getCanvasXY = function (canvas)
        {
            var x   = 0;
            var y   = 0;
            var obj = canvas;

            do {
                x += obj.offsetLeft;
                y += obj.offsetTop;

                obj = obj.offsetParent;

            } while (obj && obj.tagName.toLowerCase() != 'body');

            return [x, y];
        }



        /**
        * Registers a graph object (used when the canvas is redrawn)
        *
        * @param object obj The object to be registered
        */
        RGraph.Register = function (obj)
        {
            var key = obj.id + '_' + obj.type;

            RGraph.objects[key] = obj;
        }



        /**
        * Causes all registered objects to be redrawn
        *
        * @param string      An optional string indicating which canvas is not to be redrawn
        * @param string An optional color to use to clear the canvas
        */
        RGraph.Redraw = function ()
        {
            for (i in RGraph.objects) {
                // TODO FIXME Maybe include more intense checking for whether the object is an
RGraph object, eg obj.isRGraph == true ...?
                if (
                        typeof(i) == 'string'
&& typeof(RGraph.objects[i]) == 'object'
&& typeof(RGraph.objects[i].type) == 'string'
&& RGraph.objects[i].isRGraph)    {

                    if (!arguments[0] || arguments[0] != RGraph.objects[i].id) {
                        RGraph.Clear(RGraph.objects[i].canvas,  arguments[1]  ?  arguments[1]  :
null);
```

```
                    RGraph.objects[i].Draw();
                }
            }
        }
    }



    /**
    * Loosly mimicks the PHP function print_r();
    */
    RGraph.pr = function (obj)
    {
        var str = '';
        var indent = (arguments[2] ? arguments[2] : '');

        switch (typeof(obj)) {
            case 'number':
                if (indent == '') {
                    str+= 'Number: '
                }
                str += String(obj);
                break;

            case 'string':
                if (indent == '') {
                    str+= 'String (' + obj.length + '):'
                }
                str += '"' + String(obj) + '"';
                break;

            case 'object':
                // In case of null
                if (obj == null) {
                    str += 'null';
                    break;
                }

                str += 'Object\n' + indent + '(\n';

                for (var i=0; i<obj.length; ++i) {
                    str += indent + ' ' + i + ' => ' + RGraph.pr(obj[i], true, indent + '        ') +
'\n';
                }
```

```
                        var str = str + indent + ')';
                        break;

                    case 'function':
                        str += obj;
                        break;

                    case 'boolean':
str += 'Boolean: ' + (obj ? 'true' : 'false');
                        break;
                }

                /**
                * Finished, now either return if we're in a recursed call, or alert()
                * if we're not.
                */
                if (arguments[1]) {
                    return str;
                } else {
                    alert(str);
                }
            }


            /**
            * The RGraph registry Set() function
            *
            * @param    string name    The name of the key
            * @param    mixed    value The value to set
            * @return mixed              Returns the same value as you pass it
            */
            RGraph.Registry.Set = function (name, value)
            {
                // Store the setting
                RGraph.Registry.store[name] = value;

                // Don't really need to do this, but ho-hum
                return value;
            }


            /**
            * The RGraph registry Get() function
            *
```

```
    * @param    string name The name of the particular setting to fetch
    * @return mixed              The value if exists, null otherwise
    */
    RGraph.Registry.Get = function (name)
    {
        //return RGraph.Registry.store[name] == null ? null : RGraph.Registry.store[name];
        return RGraph.Registry.store[name];
    }


    /**
    * This function draws the background for the bar chart, line chart and scatter chart.
    *
    * @param    object obj The graph object
    */
    RGraph.background.Draw = function (obj)
    {
        var canvas    = obj.canvas;
        var context = obj.context;
        var height    = 0;
        var gutter    = obj.Get('chart.gutter');
        var variant = obj.Get('chart.variant');

        context.fillStyle = obj.Get('chart.text.color');

        // If it's a bar and 3D variant, translate
        if (variant == '3d') {
            context.save();
            context.translate(10, -5);
        }

        // X axis title
        if (typeof(obj.Get('chart.title.xaxis')) == 'string' && obj.Get('chart.title.xaxis').length) {

            var size = obj.Get('chart.text.size');
            var font = obj.Get('chart.text.font');

            context.beginPath();
            RGraph.Text(context, font, size + 2, RGraph.GetWidth(obj) / 2,
RGraph.GetHeight(obj) - (gutter * obj.Get('chart.title.xaxis.pos')), obj.Get('chart.title.xaxis'),
'center', 'center', false, false, false, true);
            context.fill();
        }
```

```
            // Y axis title
            if (typeof(obj.Get('chart.title.yaxis')) == 'string' && obj.Get('chart.title.yaxis').length) {

                var size                        = obj.Get('chart.text.size');
                var font                        = obj.Get('chart.text.font');
                var angle                       = 270;
                var yaxis_title_pos = gutter * obj.Get('chart.title.yaxis.pos');

                if (obj.Get('chart.title.yaxis.position') == 'right') {
                    angle = 90;
                    yaxis_title_pos = RGraph.GetWidth(obj) - yaxis_title_pos;
                }

                context.beginPath();
                RGraph.Text(context,
                                font,
                                size + 2,
                                yaxis_title_pos,
                                RGraph.GetHeight(obj) / 2,
                                obj.Get('chart.title.yaxis'),
                                'center',
                                'center',
                                false,
                                angle,
                                false,
                                true);
                context.fill();
            }

            obj.context.beginPath();

            // Draw the horizontal bars
            context.fillStyle = obj.Get('chart.background.barcolor1');
            height = (RGraph.GetHeight(obj) - obj.Get('chart.gutter'));

            for (var i=gutter; i < height ; i+=80) {
                obj.context.fillRect(gutter, i, RGraph.GetWidth(obj) - (gutter * 2), Math.min(40,
RGraph.GetHeight(obj) - gutter - i) );
            }

                context.fillStyle = obj.Get('chart.background.barcolor2');
                height = (RGraph.GetHeight(obj) - gutter);

                for (var i= (40 + gutter); i < height; i+=80) {
```

```
obj.context.fillRect(gutter, i, RGraph.GetWidth(obj) - (gutter * 2), i + 40 > (RGraph.GetHeight(obj)
- gutter) ? RGraph.GetHeight(obj) - (gutter + i) : 40);
                        }

                        context.stroke();



                // Draw the background grid
                if (obj.Get('chart.background.grid')) {

                        // If autofit is specified, use the .numhlines and .numvlines along with the width to
work
                        // out the hsize and vsize
                        if (obj.Get('chart.background.grid.autofit')) {

                                /**
                                * Align the grid to the tickmarks
                                */
                                if (obj.Get('chart.background.grid.autofit.align')) {

                                        // Align the horizontal lines
                                        obj.Set('chart.background.grid.autofit.numhlines',
obj.Get('chart.ylabels.count'));

                                        // Align the vertical lines for the line
                                        if (obj.type == 'line') {
                                                if (obj.Get('chart.labels') && obj.Get('chart.labels').length) {
                                                        obj.Set('chart.background.grid.autofit.numvlines',
obj.Get('chart.labels').length - 1);
                                                } else {
                                                        obj.Set('chart.background.grid.autofit.numvlines',
obj.data[0].length - 1);
                                                }

                                        // Align the vertical lines for the bar
                                        } else if (obj.type == 'bar' && obj.Get('chart.labels') &&
obj.Get('chart.labels').length) {
                                                obj.Set('chart.background.grid.autofit.numvlines',
obj.Get('chart.labels').length);
                                        }
                                }

                                var vsize = (RGraph.GetWidth(obj) - (2 * obj.Get('chart.gutter')) - (obj.type ==
'gantt' ? 2 * obj.Get('chart.gutter') : 0)) / obj.Get('chart.background.grid.autofit.numvlines');
```

```
                var hsize = (RGraph.GetHeight(obj) - (2 * obj.Get('chart.gutter'))) /
obj.Get('chart.background.grid.autofit.numhlines');

                obj.Set('chart.background.grid.vsize', vsize);
                obj.Set('chart.background.grid.hsize', hsize);
            }

            context.beginPath();
            context.lineWidth = obj.Get('chart.background.grid.width') ?
obj.Get('chart.background.grid.width') : 1;
            context.strokeStyle = obj.Get('chart.background.grid.color');

            // Draw the horizontal lines
            if (obj.Get('chart.background.grid.hlines')) {
                height = (RGraph.GetHeight(obj) - gutter)
                for (y=gutter; y < height; y+=obj.Get('chart.background.grid.hsize')) {
                    context.moveTo(gutter, y);
                    context.lineTo(RGraph.GetWidth(obj) - gutter, y);
                }

}

            if (obj.Get('chart.background.grid.vlines')) {
                // Draw the vertical lines
                var width = (RGraph.GetWidth(obj) - gutter)
for (x=gutter + (obj.type == 'gantt' ? (2 * gutter) : 0); x<=width;
x+=obj.Get('chart.background.grid.vsize')) {
                    context.moveTo(x, gutter);
                    context.lineTo(x, RGraph.GetHeight(obj) - gutter);
                }
            }

            if (obj.Get('chart.background.grid.border')) {
                // Make sure a rectangle, the same colour as the grid goes around the graph
                context.strokeStyle = obj.Get('chart.background.grid.color');
                context.strokeRect(gutter, gutter, RGraph.GetWidth(obj) - (2 * gutter),
RGraph.GetHeight(obj) - (2 * gutter));
            }
        }

        context.stroke();

        // If it's a bar and 3D variant, translate
        if (variant == '3d') {
            context.restore();
        }

        // Draw the title if one is set
        if ( typeof(obj.Get('chart.title')) == 'string') {

            if (obj.type == 'gantt') {
                gutter /= 2;
            }

            RGraph.DrawTitle(canvas,          obj.Get('chart.title'),          gutter,          null,
obj.Get('chart.text.size') + 2);
        }

        context.stroke();
    }



    /**
    * Returns the day number for a particular date. Eg 1st February would be 32
    *
    * @param      object obj A date object
    * @return     int              The day number of the given date
    */
    RGraph.GetDays = function (obj)
    {
        var year    = obj.getFullYear();
        var days    = obj.getDate();
        var month = obj.getMonth();

        if (month == 0) return days;
        if (month >= 1) days += 31;
        if (month >= 2) days += 28;

            // Leap years. Crude, but if this code is still being used
            // when it stops working, then you have my permission to shoot
            // me. Oh, you won't be able to - I'll be dead...
            if (year >= 2008 && year % 4 == 0) days += 1;

        if (month >= 3) days += 31;
        if (month >= 4) days += 30;
        if (month >= 5) days += 31;
        if (month >= 6) days += 30;
        if (month >= 7) days += 31;
        if (month >= 8) days += 31;
        if (month >= 9) days += 30;
        if (month >= 10) days += 31;
        if (month >= 11) days += 30;

        return days;
    }
    /**
    * Draws the graph key (used by various graphs)
    *
    * @param object obj The graph object
    * @param array    key An array of the texts to be listed in the key
    * @param colors An array of the colors to be used
    */
    RGraph.DrawKey = function (obj, key, colors)
    {
        var canvas    = obj.canvas;
        var context = obj.context;
        context.lineWidth = 1;

        context.beginPath();

        /**
        * Key positioned in the gutter
        */
        var keypos      = obj.Get('chart.key.position');
        var textsize = obj.Get('chart.text.size');
        var gutter      = obj.Get('chart.gutter');

        /**
        * Change the older chart.key.vpos to chart.key.position.y
        */
        if (typeof(obj.Get('chart.key.vpos')) == 'number') {
            obj.Set('chart.key.position.y', obj.Get('chart.key.vpos') * gutter);
        }

        /**
        * Account for null values in the key
        */
        var key_non_null        = [];
        var colors_non_null = [];
        for (var i=0; i<key.length; ++i) {
            if (key[i] != null) {
                colors_non_null.push(colors[i]);
```

```
                    key_non_null.push(key[i]);
                }
            }

            key        = key_non_null;
            colors = colors_non_null;



            if (keypos && keypos == 'gutter') {

                RGraph.DrawKey_gutter(obj, key, colors);


            /**
            * In-graph style key
            */
            } else if (keypos && keypos == 'graph') {

                RGraph.DrawKey_graph(obj, key, colors);

            } else {
                alert('[COMMON] (' + obj.id + ') Unknown key position: ' + keypos);
            }
        }
        /**
        * This does the actual drawing of the key when it's in the graph
        *
        * @param object obj The graph object
        * @param array    key The key items to draw
        * @param array colors An aray of colors that the key will use
        */
        RGraph.DrawKey_graph = function (obj, key, colors)
        {
            var canvas           = obj.canvas;
            var context          = obj.context;
var text_size      = typeof(obj.Get('chart.key.text.size')) == 'number' ? obj.Get('chart.key.text.size') :
obj.Get('chart.text.size');
            var text_font      = obj.Get('chart.text.font');
            var gutter           = obj.Get('chart.gutter');
var  hpos              =  obj.Get('chart.yaxispos')  ==  'right'  ?  gutter  +  10  :  RGraph.GetWidth(obj)  -
gutter - 10;
            var vpos              = gutter + 10;
            var title             = obj.Get('chart.title');
            var blob_size      = text_size; // The blob of color
            var hmargin            = 8; // This is the size of the gaps between the blob of color and the
text
            var vmargin            = 4; // This is the vertical margin of the key
            var fillstyle        = obj.Get('chart.key.background');
            var strokestyle    = 'black';
            var height              = 0;
            var width                = 0;
            obj.coordsKey = [];
            // Need to set this so that measuring the text works out OK
            context.font = text_size + 'pt ' + obj.Get('chart.text.font');
            // Work out the longest bit of text
            for (i=0; i<key.length; ++i) {
                width = Math.max(width, context.measureText(key[i]).width);
            }
            width += 5;
            width += blob_size;
            width += 5;
            width += 5;
            width += 5;
            /**
            * Now we know the width, we can move the key left more accurately
            */
            if (      obj.Get('chart.yaxispos') == 'left'
                || (obj.type == 'pie' && !obj.Get('chart.yaxispos'))
                || (obj.type == 'hbar' && !obj.Get('chart.yaxispos'))
                || (obj.type == 'hbar' && obj.Get('chart.yaxispos') == 'center')
                || (obj.type == 'rscatter' && !obj.Get('chart.yaxispos'))
                || (obj.type == 'tradar' && !obj.Get('chart.yaxispos'))
                || (obj.type == 'rose' && !obj.Get('chart.yaxispos'))
                || (obj.type == 'funnel' && !obj.Get('chart.yaxispos'))
                || (obj.type == 'vprogress' && !obj.Get('chart.yaxispos'))
                || (obj.type == 'hprogress' && !obj.Get('chart.yaxispos'))
               ) {
                hpos -= width;
            }
            /**
            * Horizontal alignment
            */
            if (typeof(obj.Get('chart.key.halign')) == 'string') {
                if (obj.Get('chart.key.halign') == 'left') {
                    hpos = gutter + 10;
                } else if (obj.Get('chart.key.halign') == 'right') {
                    hpos = obj.canvas.width - gutter    - width;
                }
            }
            /**
            * Specific location coordinates
            */
            if (typeof(obj.Get('chart.key.position.x')) == 'number') {
                hpos = obj.Get('chart.key.position.x');
            }
            if (typeof(obj.Get('chart.key.position.y')) == 'number') {
                vpos = obj.Get('chart.key.position.y');
            }
            // Stipulate the shadow for the key box
            if (obj.Get('chart.key.shadow')) {
                context.shadowColor      = obj.Get('chart.key.shadow.color');
                context.shadowBlur       = obj.Get('chart.key.shadow.blur');
                context.shadowOffsetX = obj.Get('chart.key.shadow.offsetx');
                context.shadowOffsetY = obj.Get('chart.key.shadow.offsety');
            }
            // Draw the box that the key resides in
            context.beginPath();
                context.fillStyle      = obj.Get('chart.key.background');
  context.strokeStyle = 'black';
     if (arguments[3] != false) {
                context.lineWidth = obj.Get('chart.key.linewidth') ? obj.Get('chart.key.linewidth') :
1;
                // The older square rectangled key
                if (obj.Get('chart.key.rounded') == true) {
                    context.beginPath();
                        context.strokeStyle = strokestyle;
                        RGraph.strokedCurvyRect(context, hpos, vpos, width - 5, 5 + ( (text_size
+ 5) * RGraph.getKeyLength(key)),4);
                    context.stroke();
                    context.fill();
                    RGraph.NoShadow(obj);
                } else {
                    context.strokeRect(hpos, vpos, width - 5, 5 + ( (text_size + 5) *
RGraph.getKeyLength(key)));
                    context.fillRect(hpos, vpos, width - 5, 5 + ( (text_size + 5) *
RGraph.getKeyLength(key)));
                }
            }
            RGraph.NoShadow(obj);
            context.beginPath();
                // Draw the labels given
                for (var i=key.length - 1; i>=0; i--) {
            var j = Number(i) + 1;
                    // Draw the blob of color
                    if (obj.Get('chart.key.color.shape') == 'circle') {
                        context.beginPath();
                            context.strokeStyle = 'rgba(0,0,0,0)';
                            context.fillStyle = colors[i];
                            context.arc(hpos + 5 + (blob_size / 2), vpos + (5 * j) + (text_size * j) -
text_size + (blob_size / 2), blob_size / 2, 0, 6.26, 0);
                        context.fill();

                    } else if (obj.Get('chart.key.color.shape') == 'line') {
                        context.beginPath();
                            context.strokeStyle = colors[i];
                            context.moveTo(hpos + 5, vpos + (5 * j) + (text_size * j) - text_size +
(blob_size / 2));
                            context.lineTo(hpos + blob_size + 5, vpos + (5 * j) + (text_size * j) -
text_size + (blob_size / 2));
              context.stroke();
                    } else {
                        context.fillStyle =    colors[i];
                        context.fillRect(hpos + 5, vpos + (5 * j) + (text_size * j) - text_size,
text_size, text_size + 1);
                    }

                    context.beginPath();
                    context.fillStyle = 'black';
                    RGraph.Text(context,
                                text_font,
                                text_size,
                                hpos + blob_size + 5 + 5,
                                vpos + (5 * j) + (text_size * j),
                           key[i]);
                    if (obj.Get('chart.key.interactive')) {
                        var px = hpos + 5;
                        var py = vpos + (5 * j) + (text_size * j) - text_size;
                        var pw = width - 5 - 5 - 5;
                        var ph = text_size;



                        obj.coordsKey.push([px, py, pw, ph]);
                    }
                }
            context.fill();

            /**
            * Install the interactivity event handler
            */
            if (obj.Get('chart.key.interactive')) {

                RGraph.Register(obj);

                var key_mousemove = function (e)
                {
                    var obj             = e.target.__object__;
                    var canvas          = obj.canvas;
                    var context         = obj.context;
                    var mouseCoords = RGraph.getMouseXY(e);
                    var mouseX          = mouseCoords[0];
                    var mouseY          = mouseCoords[1];

                    for (var i=0; i<obj.coordsKey.length; ++i) {

                        var px = obj.coordsKey[i][0];
                        var py = obj.coordsKey[i][1];
                        var pw = obj.coordsKey[i][2];
                        var ph = obj.coordsKey[i][3];

                        if (     mouseX > px && mouseX < (px + pw) && mouseY > py && mouseY
< (py + ph) ) {

                            // Necessary?
                            //var index = obj.coordsKey.length - i - 1;

                            canvas.style.cursor = 'pointer';

                            return;
                        }

                        canvas.style.cursor = 'default';
                    }
                }
                canvas.addEventListener('mousemove', key_mousemove, false);
                RGraph.AddEventListener(canvas.id, 'mousemove', key_mousemove);
                var key_click = function (e)
                {
                    RGraph.Redraw();

                    var obj              = e.target.__object__;
                    var canvas           = obj.canvas;
                    var context          = obj.context;
                    var mouseCoords = RGraph.getMouseXY(e);
                    var mouseX           = mouseCoords[0];
                    var mouseY           = mouseCoords[1];

                    RGraph.DrawKey(obj, obj.Get('chart.key'), obj.Get('chart.colors'));

                    for (var i=0; i<obj.coordsKey.length; ++i) {

                        var px = obj.coordsKey[i][0];
                        var py = obj.coordsKey[i][1];
                        var pw = obj.coordsKey[i][2];
                        var ph = obj.coordsKey[i][3];

                        if (    mouseX > px && mouseX < (px + pw) && mouseY > py && mouseY
< (py + ph) ) {

                            var index = obj.coordsKey.length - i - 1;

                            // HIGHLIGHT THE LINE HERE
                            context.beginPath();
                            context.strokeStyle = 'rgba(0,0,0,0.5)';
                            context.lineWidth    = obj.Get('chart.linewidth') + 2;
                            for (var j=0; j<obj.coords2[index].length; ++j) {

                                var x = obj.coords2[index][j][0];
                                var y = obj.coords2[index][j][1];

                                if (j == 0) {
                                    context.moveTo(x, y);
                                } else {
                                    context.lineTo(x, y);
                                }
                            }
                            context.stroke();


                            context.lineWidth    = 1;
                            context.beginPath();
                                context.strokeStyle = 'black';
```

```
                                        context.fillStyle       = 'white';

                                        RGraph.SetShadow(obj, 'rgba(0,0,0,0.5)', 0,0,10);

                                        context.strokeRect(px - 2, py - 2, pw + 4, ph + 4);
                                        context.fillRect(px - 2, py - 2, pw + 4, ph + 4);

                                context.stroke();
                                context.fill();


                                RGraph.NoShadow(obj);


                                context.beginPath();
                                        context.fillStyle                                                   =
obj.Get('chart.colors')[obj.Get('chart.colors').length - i - 1];
                                        context.fillRect(px, py, blob_size, blob_size);
                                context.fill();

                                context.beginPath();
                                        context.fillStyle = obj.Get('chart.text.color');

                                        RGraph.Text(context,
                                                        obj.Get('chart.text.font'),
                                                        obj.Get('chart.text.size'),
                                                        px + 5 + blob_size,
                                                        py + ph,
                                                        obj.Get('chart.key')[obj.Get('chart.key').length   -
i - 1]
                                                       );
                                context.fill();
                                canvas.style.cursor = 'pointer';
                                return;
                        }
                        canvas.style.cursor = 'default';
                }
        }
        canvas.addEventListener('click', key_click, false);
        RGraph.AddEventListener(canvas.id, 'click', key_click);


        //var key_window_click = function (e)
        //{
```

```
                    //        RGraph.Redraw();
                    //}
                    //window.addEventListener('click', key_window_click, false);
                    //RGraph.AddEventListener(canvas.id, 'window_click', key_window_click);
                }

        }
        /**
        * This does the actual drawing of the key when it's in the gutter
        *
        * @param object obj The graph object
        * @param array    key The key items to draw
        * @param array colors An aray of colors that the key will use
        */
        RGraph.DrawKey_gutter = function (obj, key, colors)
        {
                var canvas            = obj.canvas;
                var context           = obj.context;
var text_size      = typeof(obj.Get('chart.key.text.size')) == 'number' ? obj.Get('chart.key.text.size') :
obj.Get('chart.text.size');
                var text_font      = obj.Get('chart.text.font');
                var gutter            = obj.Get('chart.gutter');
                var hpos                = RGraph.GetWidth(obj) / 2;
                var vpos                = (gutter / 2) - 5;
                var title              = obj.Get('chart.title');
                var blob_size      = text_size; // The blob of color
                var hmargin            = 8; // This is the size of the gaps between the blob of color and the
text
                var vmargin            = 4; // This is the vertical margin of the key
                var fillstyle      = obj.Get('chart.key.background');
                var strokestyle = 'black';
                var length            = 0;
                // Need to work out the length of the key first
                context.font = text_size + 'pt ' + text_font;
                for (i=0; i<key.length; ++i) {
                        length += hmargin;
                        length += blob_size;
                        length += hmargin;
                        length += context.measureText(key[i]).width;
                }
                length += hmargin;
```

```
/**
* Work out hpos since in the Pie it isn't necessarily dead center
*/
if (obj.type == 'pie') {
        if (obj.Get('chart.align') == 'left') {
                var hpos = obj.radius + obj.Get('chart.gutter');

        } else if (obj.Get('chart.align') == 'right') {
                var hpos = obj.canvas.width - obj.radius - obj.Get('chart.gutter');

        } else {
                hpos = canvas.width / 2;
        }
}
/**
* This makes the key centered
*/
hpos -= (length / 2);
/**
* Override the horizontal/vertical positioning
*/
if (typeof(obj.Get('chart.key.position.x')) == 'number') {
        hpos = obj.Get('chart.key.position.x');
}
if (typeof(obj.Get('chart.key.position.y')) == 'number') {
        vpos = obj.Get('chart.key.position.y');
}
/**
* Draw the box that the key sits in
*/
if (obj.Get('chart.key.position.gutter.boxed')) {

        if (obj.Get('chart.key.shadow')) {
                context.shadowColor      = obj.Get('chart.key.shadow.color');
                context.shadowBlur        = obj.Get('chart.key.shadow.blur');
                context.shadowOffsetX = obj.Get('chart.key.shadow.offsetx');
                context.shadowOffsetY = obj.Get('chart.key.shadow.offsety');
        }


        context.beginPath();
                context.fillStyle = fillstyle;
                context.strokeStyle = strokestyle;
                if (obj.Get('chart.key.rounded')) {
                    RGraph.strokedCurvyRect(context, hpos, vpos - vmargin, length,
text_size + vmargin + vmargin)
                    // Odd... RGraph.filledCurvyRect(context, hpos, vpos - vmargin, length,
text_size + vmargin + vmargin);
                } else {
                    context.strokeRect(hpos, vpos - vmargin, length, text_size + vmargin +
vmargin);
                    context.fillRect(hpos, vpos - vmargin, length, text_size + vmargin +
vmargin);
                }

            context.stroke();
        context.fill();
            RGraph.NoShadow(obj);
        }
        /**
        * Draw the blobs of color and the text
        */
        for (var i=0, pos=hpos; i<key.length; ++i) {
            pos += hmargin;
            // Draw the blob of color - line
            if (obj.Get('chart.key.color.shape') =='line') {

                context.beginPath();
                    context.strokeStyle = colors[i];
                    context.moveTo(pos, vpos + (blob_size / 2));
                    context.lineTo(pos + blob_size, vpos + (blob_size / 2));
                context.stroke();

            // Circle
            } else if (obj.Get('chart.key.color.shape') == 'circle') {

                context.beginPath();
                    context.fillStyle = colors[i];
                    context.moveTo(pos, vpos + (blob_size / 2));
                    context.arc(pos + (blob_size / 2), vpos + (blob_size / 2), (blob_size / 2), 0,
6.28, 0);
    context.fill();
            } else {
                context.beginPath();
                    context.fillStyle = colors[i];
                    context.fillRect(pos, vpos, blob_size, blob_size);
                context.fill();
            }
            pos += blob_size;

            pos += hmargin;
            context.beginPath();
                context.fillStyle = 'black';
                RGraph.Text(context, text_font, text_size, pos, vpos + text_size - 1, key[i]);
            context.fill();
            pos += context.measureText(key[i]).width;
        }
    }
/**
* Returns the key length, but accounts for null values
*
* @param array key The key elements
*/
RGraph.getKeyLength = function (key)
{
    var len = 0;
    for (var i=0; i<key.length; ++i) {
        if (key[i] != null) {
            ++len;
        }
    }

    return len;
}
/**
* A shortcut for RGraph.pr()
*/
function pd(variable)
{
    RGraph.pr(variable);
}

function p(variable)
{
    RGraph.pr(variable);
}

/**
* A shortcut for console.log - as used by Firebug and Chromes console
*/
```

```
function cl (variable)
{
        return console.log(variable);
}
/**
* Makes a clone of an object
*
* @param obj val The object to clone
*/
RGraph.array_clone = function (obj)
{
        if(obj == null || typeof(obj) != 'object') {
                return obj;
        }
        var temp = [];
        //var temp = new obj.constructor();
        for(var i=0;i<obj.length; ++i) {
                temp[i] = RGraph.array_clone(obj[i]);
        }
        return temp;
}
/**
* This function reverses an array
*/
RGraph.array_reverse = function (arr)
{
        var newarr = [];
        for (var i=arr.length - 1; i>=0; i--) {
                newarr.push(arr[i]);
        }
        return newarr;
}
/**
* Formats a number with thousand seperators so it's easier to read
*
* @param    integer num The number to format
* @param    string          The (optional) string to prepend to the string
* @param    string          The (optional) string to ap
* pend to the string
* @return string          The formatted number
*/
RGraph.number_format = function (obj, num)
{
        var i;
var prepend = arguments[2] ? String(arguments[2]) : '';
var append    = arguments[3] ? String(arguments[3]) : '';
            var output    = '';
            var decimal = '';
var decimal_seperator    = obj.Get('chart.scale.point') ? obj.Get('chart.scale.point') : '.';
var thousand_seperator = obj.Get('chart.scale.thousand') ? obj.Get('chart.scale.thousand') : ',';
            RegExp.$1      = '';
            var i,j;
if (typeof(obj.Get('chart.scale.formatter')) == 'function') {
      return obj.Get('chart.scale.formatter')(obj, num);
}
            // Ignore the preformatted version of "1e-2"
            if (String(num).indexOf('e') > 0) {
                  return String(prepend + String(num) + append);
            }
            // We need then number as a string
            num = String(num);
            // Take off the decimal part - we re-append it later
if (num.indexOf('.') > 0) {
num          = num.replace(/\.(.*)/, '');
                  decimal = RegExp.$1;
            }
            // Thousand seperator
            //var seperator = arguments[1] ? String(arguments[1]) : ',';
            var seperator = thousand_seperator;
            /**
            * Work backwards adding the thousand seperators
            */
            var foundPoint;
            for (i=(num.length - 1),j=0; i>=0; j++,i--) {
                  var character = num.charAt(i);
                  if ( j % 3 == 0 && j != 0) {
                        output += seperator;
                  }
                  /**
                  * Build the output
                  */
                  output += character;
            }

            /**
            * Now need to reverse the string
            */
            var rev = output;
```

```
        output = '';
        for (i=(rev.length - 1); i>=0; i--) {
            output += rev.charAt(i);
        }
        // Tidy up
        output = output.replace(/^-,/, '-');
        // Reappend the decimal
        if (decimal.length) {
            output =    output + decimal_seperator + decimal;
            decimal = '';
            RegExp.$1 = '';
        }
        // Minor bugette
        if (output.charAt(0) == '-') {
            output *= -1;
            prepend = '-' + prepend;
        }
        return prepend + output + append;
    }
    /**
    * Draws horizontal coloured bars on something like the bar, line or scatter
    */
    RGraph.DrawBars = function (obj)
    {
        var hbars = obj.Get('chart.background.hbars');
        /**
        * Draws a horizontal bar
        */
       obj.context.beginPath();
        for (i=0; i<hbars.length; ++i) {

            // If null is specified as the "height", set it to the upper max value
            if (hbars[i][1] == null) {
                hbars[i][1] = obj.max;

            // If the first index plus the second index is greater than the max value, adjust
accordingly
            } else if (hbars[i][0] + hbars[i][1] > obj.max) {
                hbars[i][1] = obj.max - hbars[i][0];
            }



            // If height is negative, and the abs() value is greater than .max, use a negative
max instead
```

```
                if (Math.abs(hbars[i][1]) > obj.max) {
                    hbars[i][1] = -1 * obj.max;
            }
                // If start point is greater than max, change it to max
                if (Math.abs(hbars[i][0]) > obj.max) {
                    hbars[i][0] = obj.max;
                }

                // If start point plus height is less than negative max, use the negative max plus the
start point
                if (hbars[i][0] + hbars[i][1] < (-1 * obj.max) ) {
                    hbars[i][1] = -1 * (obj.max + hbars[i][0]);
                }
                // If the X axis is at the bottom, and a negative max is given, warn the user
                if (obj.Get('chart.xaxispos') == 'bottom' && (hbars[i][0] < 0 || (hbars[i][1] +
hbars[i][1] < 0)) ) {
                    alert('[' + obj.type.toUpperCase() + ' (ID: ' + obj.id + ') BACKGROUND HBARS]
You have a negative value in one of your background hbars values, whilst the X axis is in the
center');
                }
                var ystart = (obj.grapharea - ((hbars[i][0] / obj.max) * obj.grapharea));
                var height = (Math.min(hbars[i][1], obj.max - hbars[i][0]) / obj.max) *
obj.grapharea;
                // Account for the X axis being in the center
                if (obj.Get('chart.xaxispos') == 'center') {
                    ystart /= 2;
                    height /= 2;
                }
                ystart += obj.Get('chart.gutter')
                var x = obj.Get('chart.gutter');
                var y = ystart - height;
                var w = obj.canvas.width - (2 * obj.Get('chart.gutter'));
                var h = height;

                // Accommodate Opera :-/
                if (navigator.userAgent.indexOf('Opera') != -1 && obj.Get('chart.xaxispos') ==
'center' && h < 0) {
                    h *= -1;
                    y = y - h;
                }

                obj.context.fillStyle = hbars[i][2];
                obj.context.fillRect(x, y, w, h);
            }
```

```
        obj.context.fill();
}


/**
* Draws in-graph labels.
*
* @param object obj The graph object
*/
RGraph.DrawInGraphLabels = function (obj)
{
        var canvas    = obj.canvas;
        var context = obj.context;
        var labels    = obj.Get('chart.labels.ingraph');
        var labels_processed = [];

        // Defaults
        var fgcolor      = 'black';
        var bgcolor      = 'white';
        var direction = 1;

        if (!labels) {
                return;
        }

        /**
        * Preprocess the labels array. Numbers are expanded
        */
        for (var i=0; i<labels.length; ++i) {
                if (typeof(labels[i]) == 'number') {
                        for (var j=0; j<labels[i]; ++j) {
                                labels_processed.push(null);
                        }
                } else if (typeof(labels[i]) == 'string' || typeof(labels[i]) == 'object') {
                        labels_processed.push(labels[i]);

                } else {
                        labels_processed.push('');
                }
        }

        /**
        * Turn off any shadow
```

```
            */
            RGraph.NoShadow(obj);

            if (labels_processed && labels_processed.length > 0) {

                for (var i=0; i<labels_processed.length; ++i) {
                    if (labels_processed[i]) {
                        var coords = obj.coords[i];

                        if (coords && coords.length > 0) {
                            var  x        =  (obj.type  ==  'bar'  ?  coords[0]  +  (coords[2]  /  2)  :
coords[0]);
                            var  y        =  (obj.type  ==  'bar'  ?  coords[1]  +  (coords[3]  /  2)  :
coords[1]);
var length = typeof(labels_processed[i][4]) == 'number' ? labels_processed[i][4] : 25;

                            context.beginPath();
                            context.fillStyle     = 'black';
                            context.strokeStyle = 'black';


                            if (obj.type == 'bar') {

                                if (obj.Get('chart.variant') == 'dot') {
                                    context.moveTo(x, obj.coords[i][1] - 5);
                                    context.lineTo(x, obj.coords[i][1] - 5 - length);

                                    var text_x = x;
                                    var text_y = obj.coords[i][1] - 5 - length;

                                } else if (obj.Get('chart.variant') == 'arrow') {
                                    context.moveTo(x, obj.coords[i][1] - 5);
                                    context.lineTo(x, obj.coords[i][1] - 5 - length);

                                    var text_x = x;
                                    var text_y = obj.coords[i][1] - 5 - length;

                                } else {

                                    context.arc(x, y, 2.5, 0, 6.28, 0);
                                    context.moveTo(x, y);
                                    context.lineTo(x, y - length);

                                    var text_x = x;
```

```
        var text_y = y - length;
    }

    context.stroke();
    context.fill();


} else if (obj.type == 'line') {

    if (
        typeof(labels_processed[i]) == 'object' &&
        typeof(labels_processed[i][3]) == 'number' &&
        labels_processed[i][3] == -1
       ) {

        context.moveTo(x, y + 5);
        context.lineTo(x, y + 5 + length);

        context.stroke();
        context.beginPath();

        // This draws the arrow
        context.moveTo(x, y + 5);
        context.lineTo(x - 3, y + 10);
        context.lineTo(x + 3, y + 10);
        context.closePath();

        var text_x = x;
        var text_y = y + 5 + length;

    } else {

        var text_x = x;
        var text_y = y - 5 - length;

        context.moveTo(x, y - 5);
        context.lineTo(x, y - 5 - length);

        context.stroke();
        context.beginPath();

        // This draws the arrow
        context.moveTo(x, y - 5);
        context.lineTo(x - 3, y - 10);
```

```
                                        context.lineTo(x + 3, y - 10);
                                        context.closePath();
                                    }

                                    context.fill();
                                }



                                // Taken out on the 10th Nov 2010 - unnecessary
                                //var width = context.measureText(labels[i]).width;

                                context.beginPath();

                                    // Fore ground color
                                    context.fillStyle = (typeof(labels_processed[i]) == 'object' &&
typeof(labels_processed[i][1]) == 'string') ? labels_processed[i][1] : 'black';

                                    RGraph.Text(context,
                                                obj.Get('chart.text.font'),
                                                obj.Get('chart.text.size'),
                                                text_x,
                                                text_y,
                                                (typeof(labels_processed[i]) == 'object' &&
typeof(labels_processed[i][0]) == 'string') ? labels_processed[i][0] : labels_processed[i],
                                                'bottom',
                                                'center',
                                                true,
                                                null,
                                                (typeof(labels_processed[i]) == 'object' &&
typeof(labels_processed[i][2]) == 'string') ? labels_processed[i][2] : 'white');
                                context.fill();
                            }
                        }
                    }
                }
            }



            /**
            * This function "fills in" key missing properties that various implementations lack
            *
            * @param object e The event object
            */
            RGraph.FixEventObject = function (e)
```

```
    {
        if (RGraph.isIE8()) {

            var e = event;

            e.pageX    = (event.clientX + document.body.scrollLeft);
            e.pageY    = (event.clientY + document.body.scrollTop);
            e.target = event.srcElement;

            if (!document.body.scrollTop && document.documentElement.scrollTop) {
                e.pageX += parseInt(document.documentElement.scrollLeft);
                e.pageY += parseInt(document.documentElement.scrollTop);
            }
        }

        // This is mainly for FF which doesn't provide offsetX
        if (typeof(e.offsetX) == 'undefined' && typeof(e.offsetY) == 'undefined') {
            var coords = RGraph.getMouseXY(e);
            e.offsetX = coords[0];
            e.offsetY = coords[1];
        }

        // Any browser that doesn't implement stopPropagation() (MSIE)
        if (!e.stopPropagation) {
            e.stopPropagation = function () {window.event.cancelBubble = true;}
        }

        return e;
    }


    /**
    * Draw crosshairs if enabled
    *
    * @param object obj The graph object (from which we can get the context and canvas as
required)
    */
    RGraph.DrawCrosshairs = function (obj)
    {
        if (obj.Get('chart.crosshairs')) {
            var canvas    = obj.canvas;
            var context = obj.context;

            // 5th November 2010 - removed now that tooltips are DOM2 based.
            //if (obj.Get('chart.tooltips') && obj.Get('chart.tooltips').length > 0) {
                //alert('[' + obj.type.toUpperCase() + '] Sorry - you cannot have crosshairs
enabled with tooltips! Turning off crosshairs...');
                //obj.Set('chart.crosshairs', false);
                //return;
            //}

            canvas.onmousemove = function (e)
            {
                var e           = RGraph.FixEventObject(e);
                var canvas    = obj.canvas;
                var context = obj.context;
                var gutter    = obj.Get('chart.gutter');
                var width      = canvas.width;
                var height    = canvas.height;
                var adjustments = obj.Get('chart.tooltips.coords.adjust');

                var mouseCoords = RGraph.getMouseXY(e);
                var x = mouseCoords[0];
                var y = mouseCoords[1];

                if (typeof(adjustments) == 'object' && adjustments[0] && adjustments[1]) {
                    x = x - adjustments[0];
                    y = y - adjustments[1];
                }

                RGraph.Clear(canvas);
                obj.Draw();

                if (      x >= gutter
&& y >= gutter
&& x <= (width - gutter)
&& y <= (height - gutter)
                   ) {

                    var linewidth = obj.Get('chart.crosshairs.linewidth');
                    context.lineWidth = linewidth ? linewidth : 1;

                    context.beginPath();
                    context.strokeStyle = obj.Get('chart.crosshairs.color');

                    // Draw a top vertical line
                    context.moveTo(x, gutter);
                    context.lineTo(x, height - gutter);
                    // Draw a horizontal line
                    context.moveTo(gutter, y);
                    context.lineTo(width - gutter, y);

                    context.stroke();

                    /**
                    * Need to show the coords?
                    */
                    if (obj.Get('chart.crosshairs.coords')) {
                        if (obj.type == 'scatter') {

                            var xCoord = (((x - obj.Get('chart.gutter')) / (obj.canvas.width -
(2 * obj.Get('chart.gutter')))) * (obj.Get('chart.xmax') - obj.Get('chart.xmin'))) +
obj.Get('chart.xmin');
                                xCoord = xCoord.toFixed(obj.Get('chart.scale.decimals'));
                            var yCoord = obj.max - (((y - obj.Get('chart.gutter')) /
(obj.canvas.height - (2 * obj.Get('chart.gutter')))) * obj.max);

                                if (obj.type == 'scatter' && obj.Get('chart.xaxispos') ==
'center') {
                                    yCoord = (yCoord - (obj.max / 2)) * 2;
                                }

                                yCoord = yCoord.toFixed(obj.Get('chart.scale.decimals'));
                            var div =
RGraph.Registry.Get('chart.coordinates.coords.div');
                            var mouseCoords = RGraph.getMouseXY(e);
                            var canvasXY = RGraph.getCanvasXY(canvas);

                            if (!div) {

                                div = document.createElement('DIV');
                                div.__object__      = obj;
                                div.style.position = 'absolute';
                                div.style.backgroundColor = 'white';
                                div.style.border = '1px solid black';
                                div.style.fontFamily = 'Arial, Verdana, sans-serif';
                                div.style.fontSize = '10pt'
                                div.style.padding = '2px';
                                div.style.opacity = 1;
                                div.style.WebkitBorderRadius = '3px';
                                div.style.borderRadius = '3px';
```

```
                                        div.style.MozBorderRadius = '3px';
                                        document.body.appendChild(div);

                                        RGraph.Registry.Set('chart.coordinates.coords.div', div);
                                    }

                                    // Convert the X/Y pixel coords to correspond to the scale

                                    div.style.opacity = 1;
                                    div.style.display = 'inline';

                                    if (!obj.Get('chart.crosshairs.coords.fixed')) {
                                        div.style.left = Math.max(2, (e.pageX - div.offsetWidth -
3)) + 'px';
                                        div.style.top = Math.max(2, (e.pageY - div.offsetHeight -
3))    + 'px';
                                    } else {
                                        div.style.left = canvasXY[0] + obj.Get('chart.gutter') + 3 +
'px';
                                        div.style.top    = canvasXY[1] + obj.Get('chart.gutter') + 3 +
'px';
                                    }

                                    div.innerHTML = '<span style="color: #666">' +
obj.Get('chart.crosshairs.coords.labels.x') + ':</span> ' + xCoord + '<br><span style="color:
#666">' + obj.Get('chart.crosshairs.coords.labels.y') + ':</span> ' + yCoord;

                                    canvas.addEventListener('mouseout',
RGraph.HideCrosshairCoords, false);

                                } else {
                                    alert('[RGRAPH] Showing crosshair coordinates is only
supported on the Scatter chart');
                                }
                            }
                        } else {
                            RGraph.HideCrosshairCoords();
                        }
                    }
                }
            }

            /**
            * Thisz function hides the crosshairs coordinates
```

```
        */
        RGraph.HideCrosshairCoords = function ()
        {
                var div = RGraph.Registry.Get('chart.coordinates.coords.div');

                if (      div
&& div.style.opacity == 1
&& div.__object__.Get('chart.crosshairs.coords.fadeout')
                    ) {
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.opacity = 0.9;}, 50);
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.opacity = 0.8;}, 100);
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.opacity = 0.7;}, 150);
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.opacity = 0.6;}, 200);
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.opacity = 0.5;}, 250);
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.opacity = 0.4;}, 300);
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.opacity = 0.3;}, 350);
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.opacity = 0.2;}, 400);
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.opacity = 0.1;}, 450);
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.opacity = 0;}, 500);
                        setTimeout(function()
{RGraph.Registry.Get('chart.coordinates.coords.div').style.display = 'none';}, 550);
                }
        }



        /**
        * Trims the right hand side of a string. Removes SPACE, TAB
        * CR and LF.
        *
        * @param string str The string to trim
        */
        RGraph.rtrim = function (str)
        {
                return str.replace(/( |\n|\r|\t)+$/, '');
        }


        /**
        * Draws the3D axes/background
        */
        RGraph.Draw3DAxes = function (obj)
        {
                var gutter    = obj.Get('chart.gutter');
                var context = obj.context;
                var canvas    = obj.canvas;

                context.strokeStyle = '#aaa';
                context.fillStyle = '#ddd';

                // Draw the vertical left side
                context.beginPath();
                        context.moveTo(gutter, gutter);
                        context.lineTo(gutter + 10, gutter - 5);
                        context.lineTo(gutter + 10, canvas.height - gutter - 5);
                        context.lineTo(gutter, canvas.height - gutter);
                context.closePath();

                context.stroke();
                context.fill();

                // Draw the bottom floor
                context.beginPath();
                        context.moveTo(gutter, canvas.height - gutter);
                        context.lineTo(gutter + 10, canvas.height - gutter - 5);
                        context.lineTo(canvas.width - gutter + 10,    canvas.height - gutter - 5);
                        context.lineTo(canvas.width - gutter, canvas.height - gutter);
                context.closePath();

                context.stroke();
                context.fill();
        }

        /**
        * Turns off any shadow
        *
        * @param object obj The graph object
        */
        RGraph.NoShadow = function (obj)
```

```
{
        obj.context.shadowColor      = 'rgba(0,0,0,0)';
        obj.context.shadowBlur       = 0;
        obj.context.shadowOffsetX = 0;
        obj.context.shadowOffsetY = 0;
}


/**
* Sets the four shadow properties - a shortcut function
*
* @param object obj          Your graph object
* @param string color      The shadow color
* @param number offsetx The shadows X offset
* @param number offsety The shadows Y offset
* @param number blur        The blurring effect applied to the shadow
*/
RGraph.SetShadow = function (obj, color, offsetx, offsety, blur)
{
        obj.context.shadowColor      = color;
        obj.context.shadowOffsetX = offsetx;
        obj.context.shadowOffsetY = offsety;
        obj.context.shadowBlur       = blur;
}


/**
* This function attempts to "fill in" missing functions from the canvas
* context object. Only two at the moment - measureText() nd fillText().
*
* @param object context The canvas 2D context
*/
RGraph.OldBrowserCompat = function (context)
{
        if (!context.measureText) {

                // This emulates the measureText() function
                context.measureText = function (text)
                {
                        var textObj = document.createElement('DIV');
                        textObj.innerHTML = text;
                        textObj.style.backgroundColor = 'white';
                        textObj.style.position = 'absolute';
                        textObj.style.top = -100
                textObj.style.left = 0;
                document.body.appendChild(textObj);

                var width = {width: textObj.offsetWidth};

                textObj.style.display = 'none';

                return width;
            }
        }

        if (!context.fillText) {
            // This emulates the fillText() method
            context.fillText        = function (text, targetX, targetY)
            {
                return false;
            }
        }

        // If IE8, add addEventListener()
        if (!context.canvas.addEventListener) {
            window.addEventListener = function (ev, func, bubble)
            {
                return this.attachEvent('on' + ev, func);
            }

            context.canvas.addEventListener = function (ev, func, bubble)
            {
                return this.attachEvent('on' + ev, func);
            }
        }
    }



    /**
    * This is a function that can be used to run code asynchronously, which can
    * be used to speed up the loading of you pages.
    *
    * @param string func This is the code to run. It can also be a function pointer.
    *                         The front page graphs show this function in action. Basically
    *                        each graphs code is made in a function, and that function is
    *                        passed to this function to run asychronously.
    */
```

```
        RGraph.Async = function (func)
        {
                return setTimeout(func, arguments[1] ? arguments[1] : 1);
        }



        /**
        * A custom random number function
        *
        * @param number min The minimum that the number should be
        * @param number max The maximum that the number should be
        * @param number        How many decimal places there should be. Default for this is 0
        */
        RGraph.random = function (min, max)
        {
var dp = arguments[2] ? arguments[2] : 0;
                var r = Math.random();

                return Number((((max - min) * r) + min).toFixed(dp));
        }



        /**
        * Draws a rectangle with curvy corners
        *
        * @param context object The context
        * @param x             number The X coordinate (top left of the square)
        * @param y             number The Y coordinate (top left of the square)
        * @param w              number The width of the rectangle
        * @param h             number The height of the rectangle
        * @param                number The radius of the curved corners
        * @param                boolean Whether the top left corner is curvy
        * @param                boolean Whether the top right corner is curvy
        * @param                boolean Whether the bottom right corner is curvy
        * @param                boolean Whether the bottom left corner is curvy
        */
        RGraph.strokedCurvyRect = function (context, x, y, w, h)
        {
                // The corner radius
var r = arguments[5] ? arguments[5] : 3;

                // The corners
var corner_tl = (arguments[6] || arguments[6] == null) ? true : false;
var corner_tr = (arguments[7] || arguments[7] == null) ? true : false;
var corner_br = (arguments[8] || arguments[8] == null) ? true : false;
var corner_bl = (arguments[9] || arguments[9] == null) ? true : false;

        context.beginPath();

            // Top left side
            context.moveTo(x + (corner_tl ? r : 0), y);
            context.lineTo(x + w - (corner_tr ? r : 0), y);

            // Top right corner
            if (corner_tr) {
                context.arc(x + w - r, y + r, r, Math.PI * 1.5, Math.PI * 2, false);
            }

            // Top right side
            context.lineTo(x + w, y + h - (corner_br ? r : 0) );

            // Bottom right corner
            if (corner_br) {
                context.arc(x + w - r, y - r + h, r, Math.PI * 2, Math.PI * 0.5, false);
            }

            // Bottom right side
            context.lineTo(x + (corner_bl ? r : 0), y + h);

            // Bottom left corner
            if (corner_bl) {
                context.arc(x + r, y - r + h, r, Math.PI * 0.5, Math.PI, false);
            }

            // Bottom left side
            context.lineTo(x, y + (corner_tl ? r : 0) );

            // Top left corner
            if (corner_tl) {
                context.arc(x + r, y + r, r, Math.PI, Math.PI * 1.5, false);
            }

        context.stroke();
    }


    /**
    * Draws a filled rectangle with curvy corners
    *
    * @param context object The context
    * @param x           number The X coordinate (top left of the square)
    * @param y           number The Y coordinate (top left of the square)
    * @param w           number The width of the rectangle
    * @param h           number The height of the rectangle
    * @param             number The radius of the curved corners
    * @param             boolean Whether the top left corner is curvy
    * @param             boolean Whether the top right corner is curvy
    * @param             boolean Whether the bottom right corner is curvy
    * @param             boolean Whether the bottom left corner is curvy
    */
    RGraph.filledCurvyRect = function (context, x, y, w, h)
    {
        // The corner radius
var r = arguments[5] ? arguments[5] : 3;

        // The corners
var corner_tl = (arguments[6] || arguments[6] == null) ? true : false;
var corner_tr = (arguments[7] || arguments[7] == null) ? true : false;
var corner_br = (arguments[8] || arguments[8] == null) ? true : false;
var corner_bl = (arguments[9] || arguments[9] == null) ? true : false;

        context.beginPath();

            // First draw the corners

            // Top left corner
            if (corner_tl) {
                context.moveTo(x + r, y + r);
                context.arc(x + r, y + r, r, Math.PI, 1.5 * Math.PI, false);
            } else {
                context.fillRect(x, y, r, r);
            }

            // Top right corner
            if (corner_tr) {
                context.moveTo(x + w - r, y + r);
                context.arc(x + w - r, y + r, r, 1.5 * Math.PI, 0, false);
            } else {
                context.moveTo(x + w - r, y);
                context.fillRect(x + w - r, y, r, r);
            }
            // Bottom right corner
            if (corner_br) {
                context.moveTo(x + w - r, y + h - r);
                context.arc(x + w - r, y - r + h, r, 0, Math.PI / 2, false);
            } else {
                context.moveTo(x + w - r, y + h - r);
                context.fillRect(x + w - r, y + h - r, r, r);
            }

            // Bottom left corner
            if (corner_bl) {
                context.moveTo(x + r, y + h - r);
                context.arc(x + r, y - r + h, r, Math.PI / 2, Math.PI, false);
            } else {
                context.moveTo(x, y + h - r);
                context.fillRect(x, y + h - r, r, r);
            }

            // Now fill it in
            context.fillRect(x + r, y, w - r - r, h);
            context.fillRect(x, y + r, r + 1, h - r - r);
            context.fillRect(x + w - r - 1, y + r, r + 1, h - r - r);

        context.fill();
    }


    /**
    * A crude timing function
    *
    * @param string label The label to use for the time
    */
    RGraph.Timer = function (label)
    {
        var d = new Date();

        // This uses the Firebug console
console.log(label + ': ' + d.getSeconds() + '.' + d.getMilliseconds());
    }


    /**
    * Hides the palette if it's visible
    */
    RGraph.HidePalette = function ()
    {
        var div = RGraph.Registry.Get('palette');

        if (typeof(div) == 'object' && div) {
            div.style.visibility = 'hidden';
            div.style.display    = 'none';
            RGraph.Registry.Set('palette', null);
        }
    }


    /**
    * Hides the zoomed canvas
    */
    RGraph.HideZoomedCanvas = function ()
    {
        if (typeof(__zoomedimage__) == 'object') {
            obj = __zoomedimage__.obj;
        } else {
            return;
        }

        if (obj.Get('chart.zoom.fade.out')) {
            for (var i=10,j=1; i>=0; --i, ++j) {
                if (typeof(__zoomedimage__) == 'object') {
                    setTimeout("__zoomedimage__.style.opacity = " + String(i / 10), j * 30);
                }
            }

            if (typeof(__zoomedbackground__) == 'object') {
                setTimeout("__zoomedbackground__.style.opacity = " + String(i / 10), j * 30);
            }
        }

        if (typeof(__zoomedimage__) == 'object') {
            setTimeout("__zoomedimage__.style.display          =          'none'",
obj.Get('chart.zoom.fade.out') ? 310 : 0);
        }

        if (typeof(__zoomedbackground__) == 'object') {
            setTimeout("__zoomedbackground__.style.display          =          'none'",
obj.Get('chart.zoom.fade.out') ? 310 : 0);
```

```
        }
    }


    /**
    * Adds an event handler
    *
    * @param object obj      The graph object
    * @param string event The name of the event, eg ontooltip
    * @param object func    The callback function
    */
    RGraph.AddCustomEventListener = function (obj, name, func)
    {
        if (typeof(RGraph.events[obj.id]) == 'undefined') {
            RGraph.events[obj.id] = [];
        }

        RGraph.events[obj.id].push([obj, name, func]);

        return RGraph.events[obj.id].length - 1;
    }


    /**
    * Used to fire one of the RGraph custom events
    *
    * @param object obj      The graph object that fires the event
    * @param string event The name of the event to fire
    */
    RGraph.FireCustomEvent = function (obj, name)
    {
        if (obj && obj.isRGraph) {
            var id = obj.id;

            if (      typeof(id) == 'string'
&& typeof(RGraph.events) == 'object'
&& typeof(RGraph.events[id]) == 'object'
&& RGraph.events[id].length > 0) {

                for(var j=0; j<RGraph.events[id].length; ++j) {
                    if (RGraph.events[id][j] && RGraph.events[id][j][1] == name) {
                        RGraph.events[id][j][2](obj);
                    }
                }
            }
        }
    }


    /**
    * Checks the browser for traces of MSIE8
    */
    RGraph.isIE8 = function ()
    {
        return navigator.userAgent.indexOf('MSIE 8') > 0;
    }


    /**
    * Checks the browser for traces of MSIE9
    */
    RGraph.isIE9 = function ()
    {
        return navigator.userAgent.indexOf('MSIE 9') > 0;
    }


    /**
    * Checks the browser for traces of MSIE9
    */
    RGraph.isIE9up = function ()
    {
        navigator.userAgent.match(/MSIE (\d+)/);

        return Number(RegExp.$1) >= 9;
    }


    /**
    * This clears a canvases event handlers. Used at the start of each graphs .Draw() method.
    *
    * @param string id The ID of the canvas whose event handlers will be cleared
    */
    RGraph.ClearEventListeners = function (id)
    {
        for (var i=0; i<RGraph.Registry.Get('chart.event.handlers').length; ++i) {

            var el = RGraph.Registry.Get('chart.event.handlers')[i];
```

```
                if (el && (el[0] == id || el[0] == ('window_' + id)) ) {
                    if (el[0].substring(0, 7) == 'window_') {
                        window.removeEventListener(el[1], el[2], false);
                    } else {
                        document.getElementById(id).removeEventListener(el[1], el[2], false);
                    }

                    RGraph.Registry.Get('chart.event.handlers')[i] = null;
                }
            }
        }



        /**
        *
        */
        RGraph.AddEventListener = function (id, e, func)
        {
            RGraph.Registry.Get('chart.event.handlers').push([id, e, func]);
        }



        /**
        * This function suggests a gutter size based on the widest left label. Given that the bottom
        * labels may be longer, this may be a little out.
        *
        * @param object obj    The graph object
        * @param array    data An array of graph data
        * @return int          A suggested gutter setting
        */
        RGraph.getGutterSuggest = function (obj, data)
        {
            var                str                =                RGraph.number_format(obj,
RGraph.array_max(RGraph.getScale(RGraph.array_max(data),    obj)),    obj.Get('chart.units.pre'),
obj.Get('chart.units.post'));

            // Take into account the HBar
            if (obj.type == 'hbar') {

                var str = '';
                var len = 0;

                for (var i=0; i<obj.Get('chart.labels').length; ++i) {
```

```
                str = (obj.Get('chart.labels').length > str.length ? obj.Get('chart.labels')[i] : str);
            }
        }

        obj.context.font = obj.Get('chart.text.size') + 'pt ' + obj.Get('chart.text.font');

        len = obj.context.measureText(str).width + 5;

        return (obj.type == 'hbar' ? len / 3 : len);
    }


    /**
    * A basic Array shift gunction
    *
    * @param    object The numerical array to work on
    * @return          The new array
    */
    RGraph.array_shift = function (arr)
    {
        var ret = [];

        for (var i=1; i<arr.length; ++i) ret.push(arr[i]);

        return ret;
    }


    /**
    * If you prefer, you can use the SetConfig() method to set the configuration information
    * for your chart. You may find that setting the configuration this way eases reuse.
    *
    * @param object obj        The graph object
    * @param object config The graph configuration information
    */
    RGraph.SetConfig = function (obj, c)
    {
        for (i in c) {
            if (typeof(i) == 'string') {
                obj.Set(i, c[i]);
            }
        }

        return obj;
    }


    /**
    * This function gets the canvas height. Defaults to the actual
    * height but this can be changed by setting chart.height.
    *
    * @param object obj The graph object
    */
    RGraph.GetHeight = function (obj)
    {
        var height = obj.Get('chart.height');

return height ? height : obj.canvas.height;
    }


    /**
    * This function gets the canvas width. Defaults to the actual
    * width but this can be changed by setting chart.width.
    *
    * @param object obj The graph object
    */
    RGraph.GetWidth = function (obj)
    {

        var width = obj.Get('chart.width');

return width ? width : obj.canvas.width;
    }


    /**
    * Clears all the custom event listeners that have been registered
    *
    * @param        string Limits the clearing to this object ID
    */
    RGraph.RemoveAllCustomEventListeners = function ()
    {
        var id = arguments[0];

        if (id && RGraph.events[id]) {
            RGraph.events[id] = [];
        } else {
            RGraph.events = [];
        }
    }


    /**
    * Clears a particular custom event listener
    *
    * @param object obj The graph object
    * @param number i      This is the index that is return by .AddCustomEventListener()
    */
    RGraph.RemoveCustomEventListener = function (obj, i)
    {
        if (      typeof(RGraph.events) == 'object'
&& typeof(RGraph.events[obj.id]) == 'object'
&& typeof(RGraph.events[obj.id][i]) == 'object') {

            RGraph.events[obj.id][i] = null;
        }
    }
```

3.2.3 RGraph.pie.js 文件代码

```
    if (typeof(RGraph) == 'undefined') RGraph = {};

    /**
    * The pie chart constructor
    *
    * @param data array The data to be represented on the pie chart
    */
    RGraph.Pie = function (id, data)
    {
        this.id                   = id;
        this.canvas               = document.getElementById(id);
        this.context              = this.canvas.getContext("2d");
        this.canvas.__object__    = this;
        this.total                = 0;
        this.subTotal             = 0;
        this.angles               = [];
        this.data                 = data;
        this.properties           = [];
        this.type                 = 'pie';
        this.isRGraph          = true;


        /**
        * Compatibility with older browsers
        */
        RGraph.OldBrowserCompat(this.context);

        this.properties = {
            'chart.width':                    null,
            'chart.height':                   null,
            'chart.colors':                   ['rgb(255,0,0)', '#ddd',
'rgb(0,255,0)', 'rgb(0,0,255)', 'pink', 'yellow', '#000'],
            'chart.strokestyle':              '#999',
            'chart.linewidth':                1,
            'chart.labels':                   [],
            'chart.labels.sticks':            false,
            'chart.labels.sticks.color':      '#aaa',
            'chart.segments':                 [],
            'chart.gutter':                   25,
            'chart.title':                    '',
            'chart.title.background':         null,
            'chart.title.hpos':               null,
            'chart.title.vpos':               null,
            'chart.shadow':                   false,
            'chart.shadow.color':             'rgba(0,0,0,0.5)',
            'chart.shadow.offsetx':           3,
            'chart.shadow.offsety':           3,
            'chart.shadow.blur':              3,
            'chart.text.size':                10,
            'chart.text.color':               'black',
            'chart.text.font':                'Verdana',
            'chart.contextmenu':              null,
            'chart.tooltips':                 [],
            'chart.tooltips.event':           'onclick',
            'chart.tooltips.effect':          'fade',
            'chart.tooltips.css.class':       'RGraph_tooltip',
            'chart.tooltips.highlight':       true,
            'chart.highlight.style':          '3d',
            'chart.highlight.style.2d.fill': 'rgba(255,255,255,0.5)',
            'chart.highlight.style.2d.stroke': 'rgba(255,255,255,0)',
            'chart.radius':                   null,
            'chart.border':                   false,
            'chart.border.color':             'rgba(255,255,255,0.5)',
            'chart.key':                        null,
            'chart.key.background':             'white',
            'chart.key.position':               'graph',
            'chart.key.halign':                 'right',
            'chart.key.shadow':                 false,
            'chart.key.shadow.color':           '#666',
            'chart.key.shadow.blur':            3,
            'chart.key.shadow.offsetx':         2,
            'chart.key.shadow.offsety':         2,
            'chart.key.position.gutter.boxed': true,
            'chart.key.position.x':             null,
            'chart.key.position.y':             null,
            'chart.key.color.shape':            'square',
            'chart.key.rounded':                true,
            'chart.key.linewidth':              1,
            'chart.annotatable':                false,
            'chart.annotate.color':             'black',
            'chart.align':                      'center',
            'chart.zoom.factor':                1.5,
            'chart.zoom.fade.in':               true,
            'chart.zoom.fade.out':              true,
            'chart.zoom.hdir':                  'right',
            'chart.zoom.vdir':                  'down',
            'chart.zoom.frames':                10,
            'chart.zoom.delay':                 50,
            'chart.zoom.shadow':                true,
            'chart.zoom.mode':                  'canvas',
            'chart.zoom.thumbnail.width':       75,
            'chart.zoom.thumbnail.height':      75,
            'chart.zoom.background':            true,
            'chart.zoom.action':                'zoom',
            'chart.resizable':                  false,
            'chart.resize.handle.adjust':       [0,0],
            'chart.resize.handle.background': null,
            'chart.variant':                    'pie',
            'chart.exploded':                   []
        }

        /**
        * Calculate the total
        */
        for (var i=0,len=data.length; i<len; i++) {
            this.total += data[i];
        }
```

```
        // Check the common library has been included
        if (typeof(RGraph) == 'undefined') {
            alert('[PIE] Fatal error: The common library does not appear
to have been included');
        }
    }



    /**
    * A generic setter
    */
    RGraph.Pie.prototype.Set = function (name, value)
    {
        if (name == 'chart.highlight.style.2d.color') {
            name = 'chart.highlight.style.2d.fill';
        }

        this.properties[name] = value;
    }



    /**
    * A generic getter
    */
    RGraph.Pie.prototype.Get = function (name)
    {
        if (name == 'chart.highlight.style.2d.color') {
            name = 'chart.highlight.style.2d.fill';
        }

        return this.properties[name];
    }



    /**
    * This draws the pie chart
    */
    RGraph.Pie.prototype.Draw = function ()
    {
        /**
        * Fire the onbeforedraw event
        */
        RGraph.FireCustomEvent(this, 'onbeforedraw');
        /**
        * Reset this to an empty array
        */
        this.Set('chart.segments', []);

        /**
        * Clear all of this canvases event handlers (the ones installed
by RGraph)
        */
        RGraph.ClearEventListeners(this.id);


        this.diameter    = Math.min(RGraph.GetHeight(this),
RGraph.GetWidth(this)) - (2 * this.Get('chart.gutter'));
        this.radius      = this.Get('chart.radius') ?
this.Get('chart.radius') : this.diameter / 2;
        // this.centerx now defined below
        this.centery     = RGraph.GetHeight(this) / 2;
        this.subTotal    = 0;
        this.angles      = [];

        /**
        * Alignment (Pie is center aligned by default) Only if centerx
is not defined - donut defines the centerx
        */
        if (this.Get('chart.align') == 'left') {
            this.centerx = this.radius + this.Get('chart.gutter');

        } else if (this.Get('chart.align') == 'right') {
            this.centerx = RGraph.GetWidth(this) - (this.radius +
this.Get('chart.gutter'));

        } else {
            this.centerx = RGraph.GetWidth(this) / 2;
        }

        /**
        * Draw the shadow if required
        */
        if (this.Get('chart.shadow')) {

var offsetx = document.all ? this.Get('chart.shadow.offsetx') : 0;
var offsety = document.all ? this.Get('chart.shadow.offsety') : 0;
```

```
            this.context.beginPath();
            this.context.fillStyle = this.Get('chart.shadow.color');

            this.context.shadowColor   =
this.Get('chart.shadow.color');
            this.context.shadowBlur    =
this.Get('chart.shadow.blur');
            this.context.shadowOffsetX =
this.Get('chart.shadow.offsetx');
            this.context.shadowOffsetY =
this.Get('chart.shadow.offsety');

            this.context.arc(this.centerx + offsetx, this.centery +
offsety, this.radius, 0, 6.28, 0);

            this.context.fill();

            // Now turn off the shadow
            RGraph.NoShadow(this);
        }

        /**
        * The total of the array of values
        */
        this.total = RGraph.array_sum(this.data);

        for (var i=0,len=this.data.length; i<len; i++) {
            var angle = (this.data[i] / this.total) * 360;

            this.DrawSegment(angle,
                             this.Get('chart.colors')[i],
                             i == (this.data.length - 1),
                             i);
        }

        /**
        * Redraw the seperating lines
        */
        if (this.Get('chart.linewidth') > 0) {
            this.context.beginPath();
            this.context.lineWidth = this.Get('chart.linewidth');
            this.context.strokeStyle = this.Get('chart.strokestyle');
            for (var i=0,len=this.angles.length; i<len; ++i) {
                this.context.moveTo(this.centerx, this.centery);
                this.context.arc(this.centerx, this.centery,
this.radius, this.angles[i][0] / 57.3, (this.angles[i][0] + 0.01) / 57.3,
0);
            }

            this.context.stroke();

            /**
            * And finally redraw the border
            */
            this.context.beginPath();
            this.context.moveTo(this.centerx, this.centery);
            this.context.arc(this.centerx, this.centery, this.radius,
0, 6.28, 0);
            this.context.stroke();
        }

        /**
        * Draw label sticks
        */
        if (this.Get('chart.labels.sticks')) {
            this.DrawSticks();

            // Redraw the border going around the Pie chart if the stroke
style is NOT white
            if (
                   this.Get('chart.strokestyle') != 'white'
&& this.Get('chart.strokestyle') != '#fff'
&& this.Get('chart.strokestyle') != '#ffffff'
&& this.Get('chart.strokestyle') != 'rgb(255,255,255)'
&& this.Get('chart.strokestyle') != 'rgba(255,255,255,0)'
               ) {

                this.context.beginPath();
                    this.context.strokeStyle =
this.Get('chart.strokestyle');
                    this.context.lineWidth =
this.Get('chart.linewidth');
                    this.context.arc(this.centerx, this.centery,
this.radius, 0, 6.28, false);
                this.context.stroke();
            }
        }

        /**
        * Draw the labels
        */
        this.DrawLabels();

        /**
        * Draw the title
        */
        if (this.Get('chart.align') == 'left') {
            var centerx = this.radius + this.Get('chart.gutter');

        } else if (this.Get('chart.align') == 'right') {
            var centerx = RGraph.GetWidth(this) - (this.radius +
this.Get('chart.gutter'));

        } else {
            var centerx = null;
        }

        RGraph.DrawTitle(this.canvas, this.Get('chart.title'),
this.Get('chart.gutter'), centerx, this.Get('chart.text.size') + 2);


        /**
        * Setup the context menu if required
        */
        if (this.Get('chart.contextmenu')) {
            RGraph.ShowContext(this);
        }

        /**
        * Tooltips
        */
        if (this.Get('chart.tooltips').length) {

            /**
            * Register this object for redrawing
            */
            RGraph.Register(this);

            /**
            * The onclick event
            */
            //this.canvas.onclick = function (e)
            var canvas_onclick_func = function (e)
            {
                RGraph.HideZoomedCanvas();

                e = RGraph.FixEventObject(e);

                var mouseCoords = RGraph.getMouseXY(e);

                var canvas  = e.target;
                var context = canvas.getContext('2d');
                var obj     = e.target.__object__;



                /**
                * If it's actually a donut make sure the hyp is bigger
                * than the size of the hole in the middle
                */
                if (obj.Get('chart.variant') == 'donut' &&
Math.abs(hyp) < (obj.radius / 2)) {
                    return;
                }

                /**
                * The angles for each segment are stored in "angles",
                * so go through that checking if the mouse position
corresponds
                */
                var isDonut = obj.Get('chart.variant') == 'donut';
                var hStyle  = obj.Get('chart.highlight.style');
                var segment = obj.getSegment(e);

                if (segment) {

                    var x     = mouseCoords[0] - segment[0];
                    var y     = mouseCoords[1] - segment[1];
                    var theta = Math.atan(y / x); // RADIANS
                    var hyp   = y / Math.sin(theta);


                    if (RGraph.Registry.Get('chart.tooltip') &&
segment[5] == RGraph.Registry.Get('chart.tooltip').__index__) {
```

```
                         return;
                    } else {
                         RGraph.Redraw();
                    }


                    if (isDonut || hStyle == '2d') {

                         context.beginPath();

                         context.strokeStyle =
obj.Get('chart.highlight.style.2d.stroke');
                         context.fillStyle   =
obj.Get('chart.highlight.style.2d.fill');

                         //context.moveTo(obj.centerx, obj.centery);

                         context.moveTo(segment[0], segment[1]);
                         context.arc(segment[0], segment[1],
segment[2], RGraph.degrees2Radians(obj.angles[segment[5]][0]),
RGraph.degrees2Radians(obj.angles[segment[5]][1]), 0);
                         context.lineTo(segment[0], segment[1]);
                         context.closePath();

                         context.stroke();
                         context.fill();

                         //Removed 7th December 2010
                         //context.stroke();

                    } else {

                         context.lineWidth = 2;

                         /**
                         * Draw a white segment where the one that has been
clicked on was
                         */
                         context.fillStyle = 'white';
                         context.strokeStyle = 'white';
                         context.beginPath();
                         context.moveTo(segment[0], segment[1]);
                         context.arc(segment[0], segment[1],
segment[2], obj.angles[segment[5]][0] / 57.3, obj.angles[segment[5]][1]
/ 57.3, 0);
                         context.stroke();
                         context.fill();

                         context.lineWidth = 1;

                         context.shadowColor   = '#666';
                         context.shadowBlur    = 3;
                         context.shadowOffsetX = 3;
                         context.shadowOffsetY = 3;

                         // Draw the new segment
                         context.beginPath();
                             context.fillStyle   =
obj.Get('chart.colors')[segment[5]];
                             context.strokeStyle = 'rgba(0,0,0,0)';
                             context.moveTo(segment[0] - 3, segment[1] -
3);
                             context.arc(segment[0] - 3, segment[1] - 3,
segment[2], RGraph.degrees2Radians(obj.angles[segment[5]][0]),
RGraph.degrees2Radians(obj.angles[segment[5]][1]), 0);
                             context.lineTo(segment[0] - 3, segment[1] -
3);
                         context.closePath();

                         context.stroke();
                         context.fill();

                         // Turn off the shadow
                         RGraph.NoShadow(obj);

                         /**
                         * If a border is defined, redraw that
                         */
                         if (obj.Get('chart.border')) {
                             context.beginPath();
                             context.strokeStyle =
obj.Get('chart.border.color');
                             context.lineWidth = 5;
                             context.arc(segment[0] - 3, segment[1] - 3,
obj.radius - 2, RGraph.degrees2Radians(obj.angles[i][0]),
RGraph.degrees2Radians(obj.angles[i][1]), 0);
                             context.stroke();
                         }
```

```
                    }

                    /**
                    * If a tooltip is defined, show it
                    */

                    /**
                    * Get the tooltip text
                    */
                    if (typeof(obj.Get('chart.tooltips')) ==
'function') {
                        var text =
String(obj.Get('chart.tooltips')(segment[5]));

                    } else if (typeof(obj.Get('chart.tooltips')) ==
'object' && typeof(obj.Get('chart.tooltips')[segment[5]]) ==
'function') {
                        var text =
String(obj.Get('chart.tooltips')[segment[5]](segment[5]));

                    } else if (typeof(obj.Get('chart.tooltips')) ==
'object') {
                        var text =
String(obj.Get('chart.tooltips')[segment[5]]);

                    } else {
                        var text = '';
                    }

                    if (text) {
                        RGraph.Tooltip(canvas, text, e.pageX, e.pageY,
segment[5]);
                    }

                    /**
                    * Need to redraw the key?
                    */
                    if (obj.Get('chart.key') &&
obj.Get('chart.key').length && obj.Get('chart.key.position') ==
'graph') {
                        RGraph.DrawKey(obj, obj.Get('chart.key'),
obj.Get('chart.colors'));
                    }
                    e.stopPropagation();

                    return;
                } else if (obj.Get('chart.tooltips.event') ==
'onclick') {
                    RGraph.Redraw();
                }
            }
var event_name = this.Get('chart.tooltips.event') == 'onmousemove' ?
'mousemove' : 'click';

            this.canvas.addEventListener(event_name,
canvas_onclick_func, false);
            RGraph.AddEventListener(this.id, event_name,
canvas_onclick_func);







            /**
            * The onmousemove event for changing the cursor
            */
            //this.canvas.onmousemove = function (e)
            var canvas_onmousemove_func = function (e)
            {
                RGraph.HideZoomedCanvas();

                e = RGraph.FixEventObject(e);

                var obj     = e.target.__object__;
                var segment = obj.getSegment(e);

                if (segment) {
                    e.target.style.cursor = 'pointer';

                    return;
                }

                /**
                * Put the cursor back to null
                */
                e.target.style.cursor = 'default';
            }
            this.canvas.addEventListener('mousemove',
canvas_onmousemove_func, false);
            RGraph.AddEventListener(this.id, 'mousemove',
canvas_onmousemove_func);









            /**
            * The window onclick function
            */
            var window_onclick_func = function (e)
            {
                RGraph.HideZoomedCanvas();

                e = RGraph.FixEventObject(e);

                RGraph.Redraw();

                /**
                * Put the cursor back to null
                */
                e.target.style.cursor = 'default';
            }
            window.addEventListener('click', window_onclick_func,
false);
            RGraph.AddEventListener('window_' + this.id, 'click',
window_onclick_func);
        }


        /**
        * If a border is pecified, draw it
        */
        if (this.Get('chart.border')) {
            this.context.beginPath();
            this.context.lineWidth = 5;
            this.context.strokeStyle = this.Get('chart.border.color');
            this.context.arc(this.centerx,
                             this.centery,
                             this.radius - 2,
                             0,
                             6.28,
                             0);

            this.context.stroke();
        }

        /**
        * Draw the kay if desired
        */
        if (this.Get('chart.key') != null) {
            //this.Set('chart.key.position', 'graph');
            RGraph.DrawKey(this, this.Get('chart.key'),
this.Get('chart.colors'));
        }


        /**
        * If this is actually a donut, draw a big circle in the middle
        */
        if (this.Get('chart.variant') == 'donut') {
            this.context.beginPath();
            this.context.strokeStyle = this.Get('chart.strokestyle');
            this.context.fillStyle   =
'white';//this.Get('chart.fillstyle');
            this.context.arc(this.centerx, this.centery, this.radius /
2, 0, 6.28, 0);
            this.context.stroke();
            this.context.fill();
        }

        RGraph.NoShadow(this);

        /**
        * If the canvas is annotatable, do install the event handlers
        */
        if (this.Get('chart.annotatable')) {
            RGraph.Annotate(this);
        }
        /**
        * This bit shows the mini zoom window if requested
        */
        if (this.Get('chart.zoom.mode') == 'thumbnail' ||
this.Get('chart.zoom.mode') == 'area') {
            RGraph.ShowZoomWindow(this);
        }


        /**
        * This function enables resizing
        */
        if (this.Get('chart.resizable')) {
            RGraph.AllowResizing(this);
        }

        /**
        * Fire the RGraph ondraw event
        */
        RGraph.FireCustomEvent(this, 'ondraw');
    }


    /**
    * Draws a single segment of the pie chart
    *
    * @param int degrees The number of degrees for this segment
    */
    RGraph.Pie.prototype.DrawSegment = function (degrees, color, last,
index)
    {
        var context  = this.context;
        var canvas   = this.canvas;
        var subTotal = this.subTotal;

        context.beginPath();

            context.fillStyle   = color;
            context.strokeStyle = this.Get('chart.strokestyle');
            context.lineWidth   = 0;

            /**
            * Exploded segments
            */
            if ( (typeof(this.Get('chart.exploded')) == 'object' &&
this.Get('chart.exploded')[index] > 0)) {
                var explosion = this.Get('chart.exploded')[index];
                var x         = 0;
                var y         = 0;
                var h         = explosion;
                var t         = (subTotal + (degrees / 2)) /
(360/6.2830);
                var x         = (Math.cos(t) * explosion);
                var y         = (Math.sin(t) * explosion);

                this.context.moveTo(this.centerx + x, this.centery +
y);
            } else {
                var x = 0;
                var y = 0;
            }

            context.arc(this.centerx + x,
                        this.centery + y,
                        this.radius,
                        subTotal / 57.3,
                        (last ? 360 : subTotal + degrees) / 57.3,
                        0);

            context.lineTo(this.centerx + x, this.centery + y);

            // Keep hold of the angles
            this.angles.push([subTotal, subTotal + degrees,
this.centerx + x, this.centery + y])
        this.context.closePath();

        this.context.fill();
        //this.context.stroke();

        /**
        * Calculate the segment angle
        */
        this.Get('chart.segments').push([subTotal, subTotal +
degrees]);
        this.subTotal += degrees;
    }

    /**
    * Draws the graphs labels
    */
    RGraph.Pie.prototype.DrawLabels = function ()
    {
        var hAlignment = 'left';
        var vAlignment = 'center';
        var labels     = this.Get('chart.labels');
        var context    = this.context;

        /**
        * Turn the shadow off
        */
        RGraph.NoShadow(this);

        context.fillStyle = 'black';
        context.beginPath();

        /**
        * Draw the key (ie. the labels)
        */
        if (labels && labels.length) {

            var text_size = this.Get('chart.text.size');

            for (i=0; i<labels.length; ++i) {

                /**
                * T|his ensures that if we're given too many labels, that
we don't get an error
                */
                if (typeof(this.Get('chart.segments')[i]) ==
'undefined') {
                    continue;
                }

                // Move to the centre
                context.moveTo(this.centerx,this.centery);

                var a = this.Get('chart.segments')[i][0] +
((this.Get('chart.segments')[i][1] - this.Get('chart.segments')[i][0])
/ 2);

                /**
                * Alignment
                */
                if (a < 90) {
                    hAlignment = 'left';
                    vAlignment = 'center';
                } else if (a < 180) {
                    hAlignment = 'right';
                    vAlignment = 'center';
                } else if (a < 270) {
                    hAlignment = 'right';
                    vAlignment = 'center';
                } else if (a < 360) {
                    hAlignment = 'left';
                    vAlignment = 'center';
                }



                /**
                * Handle the additional "explosion" offset
                */
                if (typeof(this.Get('chart.exploded')) == 'object' &&
this.Get('chart.exploded')[i]) {

                    var t = ((this.angles[i][1] - this.angles[i][0]) /
2) / (360/6.2830);
                    var seperation = this.Get('chart.exploded')[i];
                    var angle = ((this.angles[i][1] -
this.angles[i][0]) / 2) + this.angles[i][0];

                    // Adjust the angles
                    var explosion_offsetx = (Math.cos(angle / 57.29) *
seperation);
                    var explosion_offsety = (Math.sin(angle / 57.29) *
seperation);
                } else {
                    var explosion_offsetx = 0;
                    var explosion_offsety = 0;
                }

                context.fillStyle = this.Get('chart.text.color');

                RGraph.Text(context,
                            this.Get('chart.text.font'),
                            text_size,
                            this.centerx + explosion_offsetx +
((this.radius + 10)* Math.cos(a / 57.3)) +
(this.Get('chart.labels.sticks') ? (a < 90 || a > 270 ? 2 : -2) : 0),
                            this.centery + explosion_offsety +
(((this.radius + 10) * Math.sin(a / 57.3))),
                            labels[i],
                            vAlignment,
                            hAlignment);
            }

            context.fill();
        }
    }



    /**
    * This function draws the pie chart sticks (for the labels)
    */
    RGraph.Pie.prototype.DrawSticks = function ()
    {
        var context  = this.context;
        var segments = this.Get('chart.segments');
        var offset   = this.Get('chart.linewidth') / 2;
        var exploded = this.Get('chart.exploded');

        for (var i=0; i<segments.length; ++i) {

            var degrees = segments[i][1] - segments[i][0];

            context.beginPath();
            context.strokeStyle =
this.Get('chart.labels.sticks.color');
            context.lineWidth   = 1;

            var midpoint = (segments[i][0] + (degrees / 2)) / 57.3;

            if (exploded && exploded[i]) {
                var extra = exploded[i];
            } else {
                var extra = 0;
            }

            context.arc(this.centerx,
                        this.centery,
                        this.radius + 7 + extra,
                        midpoint,
                        midpoint + 0.01,
                        0);


            context.arc(this.centerx,
                        this.centery,
                        this.radius - offset + extra,
                        midpoint,
                        midpoint + 0.01,
                        0);

            context.stroke();
        }
    }



    /**
    * The (now Pie chart specific) getSegment function
    *
    * @param object e The event object
    */
    RGraph.Pie.prototype.getSegment = function (e)
    {
        RGraph.FixEventObject(e);

        // The optional arg provides a way of allowing some accuracy
(pixels)
var accuracy = arguments[1] ? arguments[1] : 0;

        var obj         = e.target.__object__;
        var canvas      = obj.canvas;
        var context     = obj.context;
        var mouseCoords = RGraph.getMouseXY(e);
        var r           = obj.radius;
        var angles      = obj.angles;
        var ret         = [];
        for (var i=0; i<angles.length; ++i) {

            var x     = mouseCoords[0] - angles[i][2];
            var y     = mouseCoords[1] - angles[i][3];
            var theta = Math.atan(y / x); // RADIANS
            var hyp   = y / Math.sin(theta);
var hyp   = (hyp < 0) ? hyp + accuracy : hyp - accuracy;
```

```
            // Put theta in DEGREES
            theta *= 57.3

            /**
            * Account for the correct quadrant
            */
            if (x < 0 && y >= 0) {
                theta += 180;
            } else if (x < 0 && y < 0) {
                theta += 180;
            } else if (x > 0 && y < 0) {
                theta += 360;
            }

            if (theta > 360) {
                theta -= 360;
            }

            if (theta >= angles[i][0] && theta < angles[i][1]) {

                hyp = Math.abs(hyp);

                if (!hyp || (obj.radius && hyp > obj.radius) ) {
                    return null;
                }

                if (obj.type == 'pie' && obj.Get('chart.variant') ==
'donut' && (hyp > obj.radius || hyp < (obj.radius / 2) ) ) {
                    return null;
                }

                ret[0] = angles[i][2];
                ret[1] = angles[i][3];
ret[2] = (obj.type == 'rose') ? angles[i][2] : obj.radius;
                ret[3] = angles[i][0];
                ret[4] = angles[i][1];
                ret[5] = i;



                if (ret[3] < 0) ret[3] += 360;
                if (ret[4] > 360) ret[4] -= 360;

                return ret;
            }
        }

        return null;
    }
```

3.2.4 RGraph.common.tooltips.js 文件代码

```
    if (typeof(RGraph) == 'undefined') RGraph =
{isRGraph:true,type:'common'};

    /**
    * This is used in two functions, hence it's here
    */
    RGraph.tooltips = {};
    RGraph.tooltips.padding   = '3px';
    RGraph.tooltips.font_face = 'Tahoma';
    RGraph.tooltips.font_size = '10pt';


    /**
    * Shows a tooltip next to the mouse pointer
    *
    * @param canvas object The canvas element object
    * @param text   string The tooltip text
    * @param int      x      The X position that the tooltip should appear
at. Combined with the canvases offsetLeft
    *                        gives the absolute X position
    * @param int      y      The Y position the tooltip should appear at.
Combined with the canvases offsetTop
    *                        gives the absolute Y position
    * @param int      idx    The index of the tooltip in the graph objects
tooltip array
    */
    RGraph.Tooltip = function (canvas, text, x, y, idx)
    {
        /**
        * chart.tooltip.override allows you to totally take control of
rendering the tooltip yourself
        */
        if (typeof(canvas.__object__.Get('chart.tooltips.override'))
== 'function') {
```

```
            return
canvas.__object__.Get('chart.tooltips.override')(canvas, text, x, y,
idx);
        }

        /**
        * This facilitates the "id:xxx" format
        */
        text = RGraph.getTooltipText(text);

        /**
        * First clear any exising timers
        */
        var timers = RGraph.Registry.Get('chart.tooltip.timers');

        if (timers && timers.length) {
            for (i=0; i<timers.length; ++i) {
                clearTimeout(timers[i]);
            }
        }
        RGraph.Registry.Set('chart.tooltip.timers', []);

        /**
        * Hide the context menu if it's currently shown
        */
        if (canvas.__object__.Get('chart.contextmenu')) {
            RGraph.HideContext();
        }
        // Redraw the canvas?
        if (canvas.__object__.Get('chart.tooltips.highlight')) {
            RGraph.Redraw(canvas.id);
        }

        var effect =
canvas.__object__.Get('chart.tooltips.effect').toLowerCase();

        if (effect == 'snap' && RGraph.Registry.Get('chart.tooltip')) {

            if (   canvas.__object__.type == 'line'
                || canvas.__object__.type == 'tradar'
                || canvas.__object__.type == 'scatter'
                || canvas.__object__.type == 'rscatter'
                ) {
```

```
                var tooltipObj = RGraph.Registry.Get('chart.tooltip');

                tooltipObj.style.width  = null;
                tooltipObj.style.height = null;

                tooltipObj.innerHTML = text;
                tooltipObj.__text__  = text;

                /**
                * Now that the new content has been set, re-set the width
& height
                */
                RGraph.Registry.Get('chart.tooltip').style.width  =
RGraph.getTooltipWidth(text, canvas.__object__) + 'px';
                RGraph.Registry.Get('chart.tooltip').style.height =
RGraph.Registry.Get('chart.tooltip').offsetHeight + 'px';



                var currentx =
parseInt(RGraph.Registry.Get('chart.tooltip').style.left);
                var currenty =
parseInt(RGraph.Registry.Get('chart.tooltip').style.top);

var diffx = x - currentx - ((x +
RGraph.Registry.Get('chart.tooltip').offsetWidth) >
document.body.offsetWidth ?
RGraph.Registry.Get('chart.tooltip').offsetWidth : 0);
                var diffy = y - currenty -
RGraph.Registry.Get('chart.tooltip').offsetHeight;

                // Position the tooltip

setTimeout('RGraph.Registry.Get("chart.tooltip").style.left = "' +
(currentx + (diffx * 0.2)) + 'px"', 25);

setTimeout('RGraph.Registry.Get("chart.tooltip").style.left = "' +
(currentx + (diffx * 0.4)) + 'px"', 50);

setTimeout('RGraph.Registry.Get("chart.tooltip").style.left = "' +
(currentx + (diffx * 0.6)) + 'px"', 75);

setTimeout('RGraph.Registry.Get("chart.tooltip").style.left = "' +
(currentx + (diffx * 0.8)) + 'px"', 100);
```

```
setTimeout('RGraph.Registry.Get("chart.tooltip").style.left = "' +
(currentx + (diffx * 1.0)) + 'px"', 125);


setTimeout('RGraph.Registry.Get("chart.tooltip").style.top = "' +
(currenty + (diffy * 0.2)) + 'px"', 25);

setTimeout('RGraph.Registry.Get("chart.tooltip").style.top = "' +
(currenty + (diffy * 0.4)) + 'px"', 50);

setTimeout('RGraph.Registry.Get("chart.tooltip").style.top = "' +
(currenty + (diffy * 0.6)) + 'px"', 75);

setTimeout('RGraph.Registry.Get("chart.tooltip").style.top = "' +
(currenty + (diffy * 0.8)) + 'px"', 100);

setTimeout('RGraph.Registry.Get("chart.tooltip").style.top = "' +
(currenty + (diffy * 1.0)) + 'px"', 125);

            } else {

                alert('[TOOLTIPS] The "snap" effect is only supported on
the Line, Rscatter, Scatter and Tradar charts');
            }

            /**
            * Fire the tooltip event
            */
            RGraph.FireCustomEvent(canvas.__object__, 'ontooltip');

            return;
        }

        /**
        * Hide any currently shown tooltip
        */
        RGraph.HideTooltip();


        /**
        * Show a tool tip
        */
        var tooltipObj  = document.createElement('DIV');
        tooltipObj.className             =
canvas.__object__.Get('chart.tooltips.css.class');
        tooltipObj.style.display         = 'none';
        tooltipObj.style.position        = 'absolute';
        tooltipObj.style.left            = 0;
        tooltipObj.style.top             = 0;
        tooltipObj.style.backgroundColor = '#ffe';
        tooltipObj.style.color           = 'black';
        if (!document.all) tooltipObj.style.border = '1px solid
rgba(0,0,0,0)';
        tooltipObj.style.visibility      = 'visible';
        tooltipObj.style.paddingLeft     = RGraph.tooltips.padding;
        tooltipObj.style.paddingRight    = RGraph.tooltips.padding;
        tooltipObj.style.fontFamily      = RGraph.tooltips.font_face;
        tooltipObj.style.fontSize        = RGraph.tooltips.font_size;
        tooltipObj.style.zIndex          = 3;
        tooltipObj.style.borderRadius       = '5px';
        tooltipObj.style.MozBorderRadius    = '5px';
        tooltipObj.style.WebkitBorderRadius = '5px';
        tooltipObj.style.WebkitBoxShadow    = 'rgba(96,96,96,0.5) 3px
3px 3px';
        tooltipObj.style.MozBoxShadow       = 'rgba(96,96,96,0.5) 3px
3px 3px';
        tooltipObj.style.boxShadow          = 'rgba(96,96,96,0.5) 3px
3px 3px';
        tooltipObj.style.filter             =
'progid:DXImageTransform.Microsoft.Shadow(color=#666666,direction=135
)';
        tooltipObj.style.opacity            = 0;
        tooltipObj.style.overflow           = 'hidden';
        tooltipObj.innerHTML                = text;
        tooltipObj.__text__                 = text; // This is set
because the innerHTML can change when it's set
        tooltipObj.__canvas__               = canvas;
        tooltipObj.style.display            = 'inline';

        if (typeof(idx) == 'number') {
            tooltipObj.__index__ = idx;
        }

        document.body.appendChild(tooltipObj);

        var width  = tooltipObj.offsetWidth;
        var height = tooltipObj.offsetHeight;
        if ((y - height - 2) > 0) {
            y = y - height - 2;
        } else {
            y = y + 2;
        }

        /**
        * Set the width on the tooltip so it doesn't resize if the window
is resized
        */
        tooltipObj.style.width = width + 'px';
        //tooltipObj.style.height = 0; // Initially set the tooltip
height to nothing

        /**
        * If the mouse is towards the right of the browser window and the
tooltip would go outside of the window,
        * move it left
        */
        if ( (x + width) > document.body.offsetWidth ) {
            x = x - width - 7;
            var placementLeft = true;

            if (canvas.__object__.Get('chart.tooltips.effect') ==
'none') {
                x = x - 3;
            }

            tooltipObj.style.left = x + 'px';
            tooltipObj.style.top  = y + 'px';

        } else {
            x += 5;

            tooltipObj.style.left = x + 'px';
            tooltipObj.style.top = y + 'px';
        }


        if (effect == 'expand') {

            tooltipObj.style.left        = (x + (width / 2)) + 'px';
            tooltipObj.style.top         = (y + (height / 2)) + 'px';
            leftDelta                    = (width / 2) / 10;
            topDelta                    = (height / 2) / 10;

            tooltipObj.style.width              = 0;
            tooltipObj.style.height             = 0;
            tooltipObj.style.boxShadow          = '';
            tooltipObj.style.MozBoxShadow       = '';
            tooltipObj.style.WebkitBoxShadow    = '';
            tooltipObj.style.borderRadius       = 0;
            tooltipObj.style.MozBorderRadius    = 0;
            tooltipObj.style.WebkitBorderRadius = 0;
            tooltipObj.style.opacity = 1;

            // Progressively move the tooltip to where it should be (the
x position)

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) -
leftDelta) + 'px' }", 25));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) -
leftDelta) + 'px' }", 50));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) -
leftDelta) + 'px' }", 75));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) -
leftDelta) + 'px' }", 100));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) -
leftDelta) + 'px' }", 125));
```

```
RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) -
leftDelta) + 'px' }", 150));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) -
leftDelta) + 'px' }", 175));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) -
leftDelta) + 'px' }", 200));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) -
leftDelta) + 'px' }", 225));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) -
leftDelta) + 'px' }", 250));

            // Progressively move the tooltip to where it should be (the
Y position)

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) - topDelta)
+ 'px' }", 25));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) - topDelta)
+ 'px' }", 50));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) - topDelta)
+ 'px' }", 75));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) - topDelta)
+ 'px' }", 100));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) - topDelta)
+ 'px' }", 125));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) - topDelta)
+ 'px' }", 150));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) - topDelta)
+ 'px' }", 175));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) - topDelta)
+ 'px' }", 200));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) - topDelta)
+ 'px' }", 225));
```

```
RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) - topDelta)
+ 'px' }", 250));

            // Progressively grow the tooltip width

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 0.1)
+ "px'; }", 25));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 0.2)
+ "px'; }", 50));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 0.3)
+ "px'; }", 75));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 0.4)
+ "px'; }", 100));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 0.5)
+ "px'; }", 125));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 0.6)
+ "px'; }", 150));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 0.7)
+ "px'; }", 175));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 0.8)
+ "px'; }", 200));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 0.9)
+ "px'; }", 225));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + width +
"px'; }", 250));

            // Progressively grow the tooltip height

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
0.1) + "px'; }", 25));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
0.2) + "px'; }", 50));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
0.3) + "px'; }", 75));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
0.4) + "px'; }", 100));
RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
0.5) + "px'; }", 125));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
0.6) + "px'; }", 150));
```

```
RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
0.7) + "px'; }", 175));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
0.8) + "px'; }", 200));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
0.9) + "px'; }", 225));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + height +
"px'; }", 250));

            // When the animation is finished, set the tooltip HTML

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').innerHTML =
RGraph.Registry.Get('chart.tooltip').__text__; }", 250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.boxShadow =
'rgba(96,96,96,0.5) 3px 3px 3px'; }", 250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.MozBoxShadow =
'rgba(96,96,96,0.5) 3px 3px 3px'; }", 250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.WebkitBoxShadow =
'rgba(96,96,96,0.5) 3px 3px 3px'; }", 250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.borderRadius = '5px'; }",
250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.MozBorderRadius =
'5px'; }", 250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.WebkitBorderRadius =
'5px'; }", 250));

        } else if (effect == 'contract') {

            tooltipObj.style.left = (x - width) + 'px';
            tooltipObj.style.top  = (y - (height * 2)) + 'px';
            tooltipObj.style.cursor = 'pointer';

            leftDelta = width / 10;
            topDelta  = height / 10;

            tooltipObj.style.width  = (width * 5) + 'px';
            tooltipObj.style.height = (height * 5) + 'px';

            tooltipObj.style.opacity = 0.2;

            // Progressively move the tooltip to where it should be (the
x position)

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) +
leftDelta) + 'px' }", 25));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) +
leftDelta) + 'px' }", 50));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) +
leftDelta) + 'px' }", 75));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) +
leftDelta) + 'px' }", 100));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) +
leftDelta) + 'px' }", 125));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) +
leftDelta) + 'px' }", 150));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) +
leftDelta) + 'px' }", 175));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) +
leftDelta) + 'px' }", 200));
RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) +
leftDelta) + 'px' }", 225));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.left =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.left) +
leftDelta) + 'px' }", 250));

            // Progressively move the tooltip to where it should be (the
Y position)

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) +
(topDelta*2)) + 'px' }", 25));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) +
(topDelta*2)) + 'px' }", 50));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) +
(topDelta*2)) + 'px' }", 75));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) +
(topDelta*2)) + 'px' }", 100));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) +
(topDelta*2)) + 'px' }", 125));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) +
(topDelta*2)) + 'px' }", 150));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) +
(topDelta*2)) + 'px' }", 175));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) +
(topDelta*2)) + 'px' }", 200));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) +
(topDelta*2)) + 'px' }", 225));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.top =
(parseInt(RGraph.Registry.Get('chart.tooltip').style.top) +
(topDelta*2)) + 'px' }", 250));

            // Progressively shrink the tooltip width

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 5.5)
+ "px'; }", 25));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 5.0)
+ "px'; }", 50));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 4.5)
+ "px'; }", 75));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 4.0)
+ "px'; }", 100));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 3.5)
+ "px'; }", 125));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 3.0)
+ "px'; }", 150));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 2.5)
+ "px'; }", 175));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 2.0)
+ "px'; }", 200));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + (width * 1.5)
+ "px'; }", 225));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.width = '" + width +
"px'; }", 250));

            // Progressively shrink the tooltip height

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
5.5) + "px'; }", 25));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
5.0) + "px'; }", 50));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
4.5) + "px'; }", 75));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
4.0) + "px'; }", 100));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
3.5) + "px'; }", 125));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
3.0) + "px'; }", 150));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
2.5) + "px'; }", 175));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
2.0) + "px'; }", 200));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + (height *
1.5) + "px'; }", 225));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.height = '" + height +
"px'; }", 250));

            // When the animation is finished, set the tooltip HTML

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').innerHTML =
RGraph.Registry.Get('chart.tooltip').__text__; }", 250));
RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.boxShadow =
'rgba(96,96,96,0.5) 3px 3px 3px'; }", 250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.MozBoxShadow =
'rgba(96,96,96,0.5) 3px 3px 3px'; }", 250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.WebkitBoxShadow =
'rgba(96,96,96,0.5) 3px 3px 3px'; }", 250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.borderRadius = '5px'; }",
250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.MozBorderRadius =
'5px'; }", 250));

RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.WebkitBorderRadius =
'5px'; }", 250));

            /**
            * This resets the pointer
            */
RGraph.Registry.Get('chart.tooltip.timers').push(setTimeout("if
(RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.cursor = 'default'; }",
275));



        } else if (effect != 'fade' && effect != 'expand' && effect !=
'none' && effect != 'snap' && effect != 'contract') {
            alert('[COMMON] Unknown tooltip effect: ' + effect);
        }
        if (effect != 'none') {
            setTimeout("if (RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.opacity = 0.1;
RGraph.Registry.Get('chart.tooltip').style.border = '1px solid
rgba(96,96,96,0.2)'; }", 25);
            setTimeout("if (RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.opacity = 0.2;
RGraph.Registry.Get('chart.tooltip').style.border = '1px solid
rgba(96,96,96,0.2)'; }", 50);
            setTimeout("if (RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.opacity = 0.3;
RGraph.Registry.Get('chart.tooltip').style.border = '1px solid
rgba(96,96,96,0.2)'; }", 75);
            setTimeout("if (RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.opacity = 0.4;
RGraph.Registry.Get('chart.tooltip').style.border = '1px solid
rgba(96,96,96,0.2)'; }", 100);
            setTimeout("if (RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.opacity = 0.5;
RGraph.Registry.Get('chart.tooltip').style.border = '1px solid
rgba(96,96,96,0.2)'; }", 125);
            setTimeout("if (RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.opacity = 0.6;
RGraph.Registry.Get('chart.tooltip').style.border = '1px solid
rgba(96,96,96,0.2)'; }", 150);
            setTimeout("if (RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.opacity = 0.7;
RGraph.Registry.Get('chart.tooltip').style.border = '1px solid
rgba(96,96,96,0.4)'; }", 175);
            setTimeout("if (RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.opacity = 0.8;
RGraph.Registry.Get('chart.tooltip').style.border = '1px solid
rgba(96,96,96,0.6)'; }", 200);
            setTimeout("if (RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.opacity = 0.9;
RGraph.Registry.Get('chart.tooltip').style.border = '1px solid
rgba(96,96,96,0.8)'; }", 225);
        }

        setTimeout("if (RGraph.Registry.Get('chart.tooltip'))
{ RGraph.Registry.Get('chart.tooltip').style.opacity =
1;RGraph.Registry.Get('chart.tooltip').style.border = '1px solid
rgb(96,96,96)';}", effect == 'none' ? 50 : 250);
```

```
        /**
        * If the tooltip it self is clicked, cancel it
        */
        tooltipObj.onmousedown = function (e)
        {
            e = RGraph.FixEventObject(e)
            e.stopPropagation();
        }

        tooltipObj.onclick = function (e)
        {
            if (e.button == 0) {
                e = RGraph.FixEventObject(e);
                e.stopPropagation();
            }
        }

        /**
        * Install the function for hiding the tooltip.
        */
        document.body.onmousedown = function (event)
        {
            var tooltip = RGraph.Registry.Get('chart.tooltip');

            if (tooltip) {
                RGraph.HideTooltip();

                // Redraw if highlighting is enabled
                if
(tooltip.__canvas__.__object__.Get('chart.tooltips.highlight')) {
                    RGraph.Redraw();
                }
            }
        }

        /**
        * If the window is resized, hide the tooltip
        */
        window.onresize = function ()
        {
            var tooltip = RGraph.Registry.Get('chart.tooltip');

            if (tooltip) {
                tooltip.parentNode.removeChild(tooltip);
                tooltip.style.display = 'none';
                tooltip.style.visibility = 'hidden';
                RGraph.Registry.Set('chart.tooltip', null);

                // Redraw the graph if necessary
                if
(canvas.__object__.Get('chart.tooltips.highlight')) {
                    RGraph.Clear(canvas);
                    canvas.__object__.Draw();
                }
            }
        }

        /**
        * Keep a reference to the tooltip
        */
        RGraph.Registry.Set('chart.tooltip', tooltipObj);

        /**
        * Fire the tooltip event
        */
        RGraph.FireCustomEvent(canvas.__object__, 'ontooltip');
    }


    /**
    *
    */
    RGraph.getTooltipText = function (text)
    {
        var result = /^id:(.*)/.exec(text);

        if (result && result[1] && document.getElementById(result[1]))
{
            text = document.getElementById(result[1]).innerHTML;
        }

        return text;
    }


    /**
    *
    */
    RGraph.getTooltipWidth = function (text, obj)
    {
        var div = document.createElement('DIV');
            div.className             =
obj.Get('chart.tooltips.css.class');
            div.style.paddingLeft     = RGraph.tooltips.padding;
            div.style.paddingRight    = RGraph.tooltips.padding;
            div.style.fontFamily      = RGraph.tooltips.font_face;
            div.style.fontSize        = RGraph.tooltips.font_size;
            div.style.visibility      = 'hidden';
            div.style.position        = 'absolute';
            div.style.top            = '300px';
            div.style.left             = 0;
            div.style.display         = 'inline';
            div.innerHTML             = RGraph.getTooltipText(text);
        document.body.appendChild(div);

        return div.offsetWidth;
    }


    /**
    * Hides the currently shown tooltip
    */
    RGraph.HideTooltip = function ()
    {
        var tooltip = RGraph.Registry.Get('chart.tooltip');

        if (tooltip) {
            tooltip.parentNode.removeChild(tooltip);
            tooltip.style.display = 'none';
            tooltip.style.visibility = 'hidden';
            RGraph.Registry.Set('chart.tooltip', null);
        }
    }
```

3.3 Canvas 元素绘制项目

Canvas 元素是 HTML5 的一部分，允许脚本语言动态渲染位图像。

它最初由苹果内部使用自己 Mac OS X WebKit 推出，供应用程序使用像仪表盘的构件和 Safari 浏览器使用。

后来,有人通过Gecko内核的浏览器(尤其是Mozilla 和Firefox),Opera[1] 和 Chrome ，和超文本网络应用技术工作组建议为下一代的网络技术使用该元素。Novell 生产的 XForms 处理器插件作为 Internet Explorer 插件支持 Canvas 元素。[2] 也有人努力使用 VML 和 JavaScript 在 Internet Explorer 支持 Canvas 功能而不需要插件。[3]Google 也已开始了一个项目，使用同样的技术在 Internet Explorer 支持 Canvas 能力。[4]但 Internet Explorer 自 Internet Explorer 9 起已经可以支持 canvas 元素。

Canvas 由一个可绘制地区 HTML 代码中的属性定义决定高度和宽度。JavaScript 代码可以访问该地区，通过一套完整的绘图功能类似于其他通用二维的 API ，从而使动态生成的图形。

一些可能的用途， 包括使用 Canvas 构造图形，动画，游戏和图片。

如果您要在 html 中加入 canvas 元素，请将以下代码加入到<body>部分中：<canvas id = "canvas" width = "宽度" height = "高度">您的浏览器不支持 canvas 元素(此消息在浏览器不支持 canvas 元素时显示)</canvas>

3.3.1 使用 canvas 元素绘制美丽的花朵

HTML 代码如下

```
<!DOCTYPE html>
<meta charset="UTF-8">
<title>使用 canvas 元素绘制美丽的花朵</title>
<script type="text/javascript">
var context;
var A,n;
function btn_onclick()
{
    var width;
    var Height;
    var canvas;
    var Xo,Yo;
    var k;
    canvas=document.getElementById("canvas");
    width=canvas.width;
    height=canvas.height;
    context=canvas.getContext('2d');
    Xo=width/2;
    Yo=height/2;
    k=parseInt(document.getElementById("drawType").value);
    if(k==2)
        A=Yo*0.25;
    else
        A=Yo*0.75;
    context.save();//保存当前绘制状态
    context.clearRect(0,0,width,height);//擦除之前绘制的图形
    context.translate(Xo,Yo);//坐标原点移动到 canvas 元素中央
    context.beginPath();//开始创建路径
    for(var B=0;B<=6.28;B=B+0.01)
    {
         draw(B);//绘制花朵曲线
    }
    context.closePath();//关闭路径
    context.restore();//恢复坐标轴平移之前的绘制状态
}
function draw(B)
{
    var n=10;
    switch(parseInt(document.getElementById("drawType").value))
    {
         case 3://大丽花
             r=A*Math.sin(n*B)*Math.exp(-B/(20));
             break;
         case 2://令箭荷花
```

```
                r=A*(Math.sin(n*B)+3*Math.sin(3*n*B));
                break;
            case 1://蓬莱菊
                r=A*Math.sin(n*B);
        }

        //极坐标的直角坐标
        x=r*Math.cos(B);
        y=r*Math.sin(B);

        context.fillStyle="green";//设置填充颜色
        context.strokeStyle="black";//设置边框颜色
        context.lineTo(-x,-y);//绘制直线
        context.fill();//填充图形
        context.stroke();//绘制边框
    }
    </script>
    <h1>使用 canvas 元素绘制美丽的花朵</h1>
    花的类型：
    <select id="drawType">
    <option value="1">蓬莱菊</option>
    <option value="2">令箭荷花</option>
    <option value="3">大丽花</option>
    </select>
    <input    type="button"    id="btn"    value="    绘    制    "
onclick="btn_onclick()"/><br/>
    <canvas id="canvas" width="200px" height="200px"></canvas>
```

3.3.2 绘制指针式动画时钟

```
    <!DOCTYPE html>
    <head>
    <meta charset="UTF-8">
    <title>使用 canvas 元素绘制指针式动画时钟</title>
    <script type="text/javascript">
    var canvas;
    var context;
    //页面装载
    function window_onload()
    {
        canvas=document.getElementById("canvas");//获取 canvas 元素
        context=canvas.getContext('2d');//获取 canvas 元素的图形上下文
对象
        setInterval("draw()",1000);//每隔一秒重绘时钟，重新显示时间
    }
    //绘制时钟
    function draw()
    {
        var radius=Math.min(canvas.width / 2, canvas.height / 2) -25;//
时钟罗盘半径
        var centerx=canvas.width/2;//时钟中心横坐标
        var centery=canvas.height/2;//时钟中心纵坐标
        context.clearRect(0,0,canvas.width,canvas.height);//擦除之前
所绘时钟

        context.save();//保存当前绘制状态

        //绘制时钟圆盘
        context.fillStyle = '#efefef';//时钟背景色
        context.strokeStyle = '#c0c0c0';//时钟边框颜色
        context.beginPath();//开始创建路径
        context.arc(centerx,centery,radius, 0,Math.PI*2, 0);//创建圆形
罗盘路径
        context.fill();//用背景色填充罗盘
        context.stroke();//用边框颜色绘制罗盘边框
        context.closePath();//关闭路径
        context.restore();//恢复之前保存的绘制状态
        //绘制时钟上表示小时的文字
        var r = radius - 10;//缩小半径，因为要将文字绘制在时钟内部
        context.font= 'bold 16px 宋体';//指定文字字体
        Drawtext('1', centerx + (0.5 * r), centery - (0.88 * r));
        Drawtext('2', centerx + (0.866 * r), centery - (0.5 * r));
        Drawtext('3', centerx + radius - 10,centery);
        Drawtext('4', centerx + (0.866 * r), centery + (0.5 * r));
        Drawtext('5', centerx + (0.5 * r), centery + (0.866 * r));
        Drawtext('6', centerx, centery + r);
        Drawtext('7', centerx - (0.5 * r), centery + (0.866 * r));
        Drawtext('8', centerx - (0.866 * r), centery + (0.49 * r));
        Drawtext('9', centerx - radius + 10, centery);
        Drawtext('10',centerx - (0.866 * r),centery - (0.50 * r));
        Drawtext('11', centerx - (0.51 * r), centery - (0.88 * r));
        Drawtext('12', centerx, 35);

        //绘制时钟指针
        var date=new Date();//获取需要表示的时间
        var h = date.getHours();//获取当前小时
        var m = date.getMinutes();//获取当前分钟
        var s=date.getSeconds();//获取当前秒
        var a = ((h/12) *Math.PI*2) - 1.57 + ((m / 60) * 0.524);//根据
当前时间计算指针角度

        context.save();//保存当前绘制状态
        context.fillStyle='black'; //指定指针中心点的颜色
        context.beginPath();//开始创建路径
        context.arc(centerx,centery,3,0,Math.PI * 2, 0);//创建指针中心
点的路径
        context.closePath();//关闭路径
        context.fill();//填充指针中心点

        context.lineWidth=3;//指定指针宽度
        context.fillStyle='darkgray';//指定指针填充颜色
        context.strokeStyle='darkgray';//指定指针边框颜色
        context.beginPath();//开始创建路径
        //绘制小时指针
        context.arc(centerx,centery,radius - 55, a + 0.01, a, 1);
        context.lineTo(centerx,centery);
        //绘制分钟指针
        context.arc(centerx,centery,radius - 40, ((m/60) * 6.27) - 1.57,
((m/60) * 6.28) - 1.57, 0);
        context.lineTo(canvas.width / 2, canvas.height / 2);
        //绘制秒钟指针
        context.arc(centerx,centery,radius - 30, ((s/60) * 6.27) - 1.57,
((s/60) * 6.28) - 1.57, 0);
        context.lineTo(centerx,centery);
        context.closePath();//关闭路径
        context.fill();//填充指针
        context.stroke();//绘制指针边框
        context.restore();//恢复之前保存的绘制状态

        //指定时钟下部当前时间所用的字符串，文字格式为hh:mm:dd
        var hours   = String(h);
        var minutes = String(m);
        var seconds = String(s);

        if (hours.length == 1)    h   = '0' + h;
        if (minutes.length == 1) m = '0' + m;
        if (seconds.length == 1) s = '0' + s;

        var str =h + ':' + m + ':' +s;
        //绘制时钟下部的当前时间
        Drawtext(str, centerx, centery + radius + 12);

    }
    function Drawtext(text, x, y)
    {
        //因为需要使用到坐标平移，所以在平移前线保存当前绘制状态
        context.save();

        x -= (context.measureText(text).width / 2);//文字起点横坐标
        y +=9;//文字起点纵坐标

        context.beginPath();//开始创建路径
        context.translate(x, y);//平移坐标
        context.fillText(text,0,0);//填充文字
        context.restore();
    }

    </script>
    <style>
    div{
            display: -moz-box;
            display: -webkit-box;
            -moz-box-pack: center;
            -webkit-box-pack: center;
      width:100%;
    }
    canvas{
        background-color:white;
    }
    </style>
    </head>
    <body onload="window_onload()">
    <div><h1>使用 canvas 元素绘制指针式动画时钟</h1></div>
    <div><canvas                id="canvas"              width="200px"
height="200px"></canvas><div>
    </body>
    </html>
```

3.3.3 小球弹跳游戏

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8">
<title>小球弹跳游戏</title>
<script type="text/javascript">
var BallX,BallY; //小球在 canvas 元素中的横坐标与纵坐标
var AddX,AddY; //小球每次移动时的横向移动距离与纵向移动距离
var width,height;//canvas 元素的宽度与高度
var canvas;//canvas 元素
var context;//canvas 元素的图形上下文对象
var functionId;//用来停止动画函数的整型变量
//点击开始游戏按钮
function btnBegin_onclick()
{
canvas=document.getElementById("canvas");//获取 canvas 元素
width=canvas.width;//获取 canvas 元素的宽度
height=canvas.height;//获取 canvas 元素的高度
context=canvas.getContext('2d'); //获取 canvas 元素的图形上下文对象
BallX=parseInt(Math.random()*canvas.width);//随机设置小球的当前横坐
标
BallY=parseInt(Math.random()*canvas.height);//随机设置小球的当前纵
坐标
AddX=-5;//设置小球每次横向移动距离为 5
AddY=-5;//设置小球每次纵向移动距离为 5
draw();//绘制矩形桌面与小球
    //使开始游戏按钮变为无效
document.getElementById("btnBegin").disabled="disabled";
//每 0.1 秒重绘矩形桌面与小球，改变小球位置以产生动画效果
functionId=setInterval("draw()",100);
}
//重绘矩形桌面与小球
function draw()
{
context.clearRect(0,0,width,height);//清除 canvas 元素中的内容
context.save();//保存当前绘制状态
context.fillStyle="lightgreen"; //设置桌面为淡绿色
context.strokeStyle="black";//设置桌面边框为黑色
context.linewidth=3; //设置桌面边框宽度
context.fillRect(3,3,width-5,height-5);//绘制淡绿色桌面
context.strokeRect(3,3,width-5,height-5);//绘制桌面黑色边框。
context.beginPath();//开始创建路径
context.fillStyle="blue";//设置小球为蓝色
context.arc(BallX,BallY,5,0,Math.PI * 2,false);//创建小球路径
BallX+=AddX;//计算小球移动后的下次绘制时的横坐标
BallY+=AddY;//计算小球移动后的下次绘制时的纵坐标
if(BallX<5)//小球向左移动时位置超过左边框
{
BallX=5;//将小球移到桌面内
AddX=-AddX;//改变小球移动方向，使其向右移动
}
else if(BallX>width-5)//小球向右移动时位置超过右边框
{
BallX=width-5;//将小球移到桌面内
AddX=-AddX;//改变小球移动方向，使其向左移动
}
if(BallY<5)//小球向上移动时位置超过上边框
{
BallY=5;//将小球移到桌面内
AddY=-AddY;//改变小球移动方向，使其向下移动
}
else if(BallY>height-5)//小球向下移动时位置超过下边框
{
BallY=height-5;//将小球移到桌面内
AddY=-AddY;//改变小球移动方向，使其向上移动
}
context.closePath();//关闭路径
context.fill(); //绘制小球
context.restore();//恢复上次保存的绘制状态
}
function canvas_mouseup(ev)
{
var differenceX;//鼠标击中点与小球中心点的横向偏差
var differenceY; //鼠标击中点与小球中心点的纵向偏差
//计算鼠标击中点与小球中心点的横向偏差
differenceX=ev.pageX-document.getElementById("canvas").offsetLeft-B
allX;
//计算鼠标击中点与小球中心点的纵向偏差
differenceY=ev.pageY-document.getElementById("canvas").offsetTop-Ba
llY;
//如果横向偏差与纵向偏差均在 5 个像素之内即为击中小球，因为小球的半径
为 5
if(-5<=differenceX&&differenceX<=5)
if(-5<=differenceY&&differenceY<=5)
{
alert("恭喜您获胜！游戏结束");
clearInterval(functionId);//停止动画
//恢复开始游戏按钮为有效状态
document.getElementById("btnBegin").disabled="";
}
}
//画面打开时添加鼠标点击 canvas 元素时的事件处理
function window_onload()
{
document.getElementById("canvas").onmouseup=canvas_mouseup;
}
</script>
</head>
<body onload="window_onload()">
<h1>小球弹跳游戏</h1>
<input type="button" id="btnBegin" value="开始游戏"
onclick="btnBegin_onclick()"/><br/>
<canvas id="canvas" width=400px height=200px></canvas>
</body>
</html>
```

3.3.4 对图像使用放大镜

```
<!DOCTYPE html>
<head>
```

```
<meta charset="UTF-8">
<title>对图像使用放大镜</title>
<script type="text/javascript">
function window_onload()
{
var canvas1 = document.getElementById('canvas1');//获取显示原图的
canvas 元素
if (canvas1 == null)
return false;
context = canvas1.getContext('2d'); //获取显示原图的 canvas 元素的图
形上下文对象
//获取图像源
var image = new Image();
image.src = "tyl.jpg";
//绘制原图
image.onload=function(){
context.drawImage(image,0,0);
}
canvas1.onmousemove=canvas1_onmouse_move;//添加原图像获取鼠标焦点时
的处理函数
canvas1.onmouseout=canvas1_onmouse_out;//添加原图像失去鼠标焦点时的
处理函数
}
//原图像获取鼠标焦点时的处理函数
function canvas1_onmouse_move(ev)
{
var canvas1,canvas2;//原图像使用的 canvas 元素与放大镜中图像使用的
canvas 元素
var x,y;//鼠标在 canvas 元素中的相对坐标点
var drawWidth,drawHeight;//鼠标所指区域的宽度与高度
canvas1=document.getElementById("canvas1");//获取原图像使用的 canvas
元素
canvas2=document.getElementById("canvas2");//获取放大镜中图像使用的
canvas 元素
var context = canvas2.getContext('2d'); //获取放大镜中图像使用的
canvas 元素的图形上下文对象
canvas2.style.display="inline"; //显示放大镜
context.clearRect(0,0,canvas2.width,canvas2.height);//擦除放大镜中
原图像
x=ev.pageX-canvas1.offsetLeft+2;//鼠标在 canvas 元素中 X 轴上的相对坐标点+2,+2 是为了避免鼠标移动到放大镜上
y=ev.pageY-canvas1.offsetTop+2;//鼠标在 canvas 元素中 Y 轴上的相对坐标点+2,+2 是为了避免鼠标移动到放大镜上
canvas2.style.left=(ev.pageX+2)+"px";//设置放大镜在原图上的 X 轴上的坐标点
canvas2.style.top=(ev.pageY+2)+"px";//设置放大镜在原图上的 Y 轴上的坐标点
//获取放大镜图像的图像源
var image = new Image();
image.src = "ty1.jpg";
//获取鼠标所指区域的宽度
if(x+40>canvas1.width)//如果鼠标所指区域的宽度超出原图宽度
drawWidth=canvas1.width-x;//设置鼠标所指区域宽度为原图中剩余宽度
else
drawWidth=40;//设置鼠标所指区域的宽度为 40 像素
//获取鼠标所指区域的高度
if(y+40>canvas1.height)//如果鼠标所指区域的高度超出原图高度
drawHeight=canvas1.height-y;//设置鼠标所指区域高度为原图中剩余高度
else
drawHeight=40;//设置鼠标所指区域的高度为 40 像素
//放大 2 倍绘制放大镜图像
context.drawImage(image,x,y,drawWidth,drawHeight,0,0,drawWidth*2,drawHeight*2);
}
//鼠标移出原图像外
function canvas1_onmouse_out()
{
var canvas2=document.getElementById("canvas2");//获取放大镜所用 canvas 元素
//重置 canvas 元素的位置
canvas2.style.left="0px";
canvas2.style.top="0px";
//隐藏放大镜
canvas2.style.display="none";
}
</script>
```

```
<style>
canvas{
background-color:white;
position:absolute;
}
canvas#canvas1{
z-index:1;
}
canvas#canvas2{
z-index:2;
left:0px;
top:0px;
border:thin dashed black;
border-radius: 40px;
-moz-border-radius: 40px;
-o-border-radius: 40px;
-webkit-border-radius: 40px;
display:none;
}
</style>
</head>
<body onload="window_onload()">
<article>
<h1>对图像使用放大镜</h1>
<canvas id="canvas1" width="100px" height="130px"></canvas>
<canvas id="canvas2" width="80px" height="82px"></canvas>
</article>
</body>
</html>
```

3.3.5 动画的形式装载图像

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8">
```

```
    <title>用动画的形式装载图像</title>
    <script type="text/javascript">
    var width,height;
    var context,image,functionId;
    var drawLeft,drawWidth;
    var drawTop,drawHeight;
    var spaceWidth,spaceHeight;
    function window_onload()
    {
         var canvas = document.getElementById('canvas');
         context = canvas.getContext('2d');
         image = new Image();
         image.src = "tyl.jpg";
         width=canvas.width;
         height=canvas.height;
    }
    function btn1_onclick()
    {
        context.fillStyle = "#EEEEFF";
        context.fillRect(0, 0,width,height);
        drawWidth=0;
        functionId=self.setInterval("drawImg1()",100);
        btnDisable();
    }
    function drawImg1()
    {
    context.drawImage(image,0,0,drawWidth,image.height,0,0,drawWidth,
image.height);
        drawWidth=drawWidth+2;
        if(drawWidth>width)
        {
            window.clearInterval(functionId);
            btnEnable();
        }
    }
    function btn2_onclick()
    {
        context.fillStyle = "#EEEEFF";
        context.fillRect(0, 0,width,height);
        drawHeight=0;
        functionId=self.setInterval("drawImg2()",100);
        btnDisable();
    }
    function drawImg2()
    {
        context.save();

context.drawImage(image,0,0,image.width,drawHeight,0,0,image.width,dr
awHeight);
        drawHeight=drawHeight+2;
        if(drawHeight>height)
        {
            window.clearInterval(functionId);
            btnEnable();
        }
        context.restore();
    }
    function btn3_onclick()
    {
        context.fillStyle = "#EEEEFF";
        context.fillRect(0, 0,width,height);
        drawLeft=width/2;
        drawWidth=0;
        functionId=self.setInterval("drawImg3()",100);
        btnDisable();
    }
    function drawImg3()
    {
        context.save();

context.drawImage(image,drawLeft,0,drawWidth,image.height,drawLeft,0,
drawWidth,image.height);
        drawLeft=drawLeft-1;
        drawWidth=drawWidth+2;
        if(drawLeft<=0)
        {
            window.clearInterval(functionId);
            btnEnable();
        }
        context.restore();
    }
    function btn4_onclick()
    {
        context.fillStyle = "#EEEEFF";
        context.fillRect(0, 0,width,height);
        drawTop=height/2;
        drawHeight=0;
        functionId=self.setInterval("drawImg4()",100);
        btnDisable();
    }
    function drawImg4()
    {
        context.save();
    context.drawImage(image,0,drawTop,image.width,drawHeight,0,drawTo
p,image.width,drawHeight);
        drawTop=drawTop-1;
        drawHeight=drawHeight+2;
        if(drawTop<=0)
        {
            window.clearInterval(functionId);
            btnEnable();
        }
        context.restore();
    }
    function btn5_onclick()
    {
        context.fillStyle = "#EEEEFF";
        context.fillRect(0, 0,width,height);
        spaceWidth=width/10;
        drawWidth=0;
        functionId=self.setInterval("drawImg5()",100);
        btnDisable();
    }
    function drawImg5()
    {
         for(i=0;i<10;i++)
         {

context.drawImage(image,0+i*spaceWidth,0,drawWidth,image.height,0+i*s
paceWidth,0,drawWidth,image.height);
         }

        drawWidth+=1;

        if(drawWidth>spaceWidth)
        {
           window.clearInterval(functionId);
           btnEnable();
        }
    }
    function btn6_onclick()
    {
        context.fillStyle = "#EEEEFF";
        context.fillRect(0, 0,width,height);
        spaceHeight=height/10;
        drawHeight=0;
        functionId=self.setInterval("drawImg6()",100);
        btnDisable();
    }
    function drawImg6()
    {
         context.save();
         context.clearRect(0, 0,width,height);
         for(i=0;i<10;i++)
         {

context.drawImage(image,0,0+i*spaceHeight,image.width,drawHeight,0,0+
i*spaceHeight,image.width,drawHeight);
         }

        drawHeight+=1;

        if(drawHeight>spaceHeight)
        {
            window.clearInterval(functionId);
            btnEnable();
        }
        context.restore();
    }
    function btnDisable()
    {
        document.getElementById("btn1").disabled="disabled";
        document.getElementById("btn2").disabled="disabled";
        document.getElementById("btn3").disabled="disabled";
        document.getElementById("btn4").disabled="disabled";
        document.getElementById("btn5").disabled="disabled";
        document.getElementById("btn6").disabled="disabled";
    }
    function btnEnable()
    {
        document.getElementById("btn1").disabled="";
        document.getElementById("btn2").disabled="";
        document.getElementById("btn3").disabled="";
        document.getElementById("btn4").disabled="";
        document.getElementById("btn5").disabled="";
        document.getElementById("btn6").disabled="";
    }
    </script>
    <style>
    article{
        align:center;
    }
    canvas{
        background-color:white;
    }
    div#divLeft{
        width:150px;
        float:left;
    }
    div#divRight{
        float:left;
    }
    input[type='button']{
        width:140px;

    }
    </style>
    </head>
    <body onload="window_onload()">
    <article>
    <h1>用动画的形式装载图像</h1>
    <div id="divLeft">
    <input  type="button"  id="btn1"  value=" 从 左 往 右 装 载 "
onclick="btn1_onclick()"></button>
```

```
    <input  type="button"  id="btn2"  value=" 从 上 往 下 装 载 "
onclick="btn2_onclick()"></button>
    <input  type="button"  id="btn3"  value=" 横 向 窗 帘 式 拉 开 "
onclick="btn3_onclick()"></button>
    <input  type="button"  id="btn4"  value=" 竖 向 窗 帘 式 拉 开 "
onclick="btn4_onclick()"></button>
    <input  type="button"  id="btn5"  value=" 横 向 百 叶 窗 式 展 开 "
onclick="btn5_onclick()"></button>
    <input  type="button"  id="btn6"  value=" 纵 向 百 叶 窗 式 展 开 "
onclick="btn6_onclick()"></button>
    </div>
    <div id="divRight">
    <canvas id="canvas" width="100" height="130"></canvas>
    </div>
    </article>
    </body>
    </html>
```

3.3.6 彩色照片转换成黑白照片

```
    <!DOCTYPE html>
    <html>
    <head>
    <script type="text/javascript">
    var canvas,ctx;
    function btnConvert_onclick() {
        canvas = document.getElementById("myCanvas");
        var imgElement=document.getElementById("img");
        canvas.width=imgElement.width;
        canvas.height=imgElement.height;
        ctx = canvas.getContext("2d");
        imgElement.onload = function() {
            ctx.drawImage(imgElement, 0, 0);
            imageConvertToGray();
        }
        imgElement.src = "tyl.jpg";
        document.getElementById("btnSave").disabled="";
    }
    function imageConvertToGray() {
        var length = canvas.width * canvas.height;
        imageData    =    ctx.getImageData(0,    0,    canvas.width,
canvas.height);
        for (var i = 0; i < length * 4; i += 4) {
            var myRed = imageData.data[i];
            var myGreen = imageData.data[i + 1];
            var myBlue = imageData.data[i + 2];
            myGray = parseInt((myRed + myGreen + myBlue) / 3);
            imageData.data[i] = myGray;
            imageData.data[i + 1] = myGray;
            imageData.data[i + 2] = myGray;
        }
        ctx.putImageData(imageData, 0, 0);
    }
    function btnSave_onclick()
    {
        window.location =canvas.toDataURL("image/jpeg");
    }
    </script>
    </head>
    <body>
    <h1>将彩色照片转换成黑白照片</h1>
    <img id="img" src="tyl.jpg"><br/>
    <input   type="button"   id="btnConvert"   value="  转  换  "
onclick="btnConvert_onclick();"/><input  type="button"  id="btnSave"
value="保存图片" onclick="btnSave_onclick();" disabled/><br/>
    <canvas id="myCanvas" width="200" height="200"/>
    </body>
    </html>
```