

Flash AS3 实现黑白棋游戏


潘惠勇, 夏敏捷

(中原工学院, 郑州 450007)

摘要: 采用 Flash AS3 技术实现黑白棋游戏。阐述游戏中翻转对方的棋子、输赢判断等实现和制作棋子影片剪辑元件等关键问题。

关键词: 黑白棋; Flash AS3 技术; 翻转棋子

1 引言

黑白棋, 又叫反棋 (Reversi)、奥赛罗棋 (Othello)、苹果棋、翻转棋。黑白棋在西方和日本很流行。黑白棋的棋盘是一个有 8*8 方格的棋盘。开始时在棋盘正中有两白两黑 4 个棋子交叉放置, 黑棋总是先下子。游戏通过相互翻转对方的棋子 (当自己放下的棋子在横、竖、斜 8 个方向内有一个自己的棋子, 则被夹在中间的全部翻转会成为自己的棋子), 如果玩家在棋盘上没有地方可以下子, 则该玩家对手可以连下。最后以棋盘上谁的棋子多来判断胜负。本文使用 Flash AS3 开发黑白棋游戏程序。该游戏具有显示执棋方可以落棋子的位置提示功能和判断胜负功能。在游戏过程中, 点击“帮助”按钮则显示执棋方可落子位置 (图片  表示可落子位置)。游戏运行界面如图 1 所示。

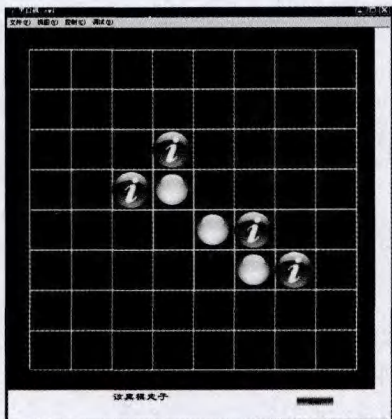



图 1 i 表示执棋方 (白方) 可落子位置

2 黑白棋游戏设计思路

2.1 棋子和棋盘

游戏开发时, 需要事先准备黑白两色棋子 (如图 2 所示) 和棋盘图片, 这里设计棋子影片剪辑使用图 2 中 4 张图, 黑白两色棋子各一帧, 图片  (表示可落子位置) 第 3 帧, 游戏背景方格是第 4 帧。游戏最初显示时, 棋盘上布满所有 64 个

棋子影片剪辑 (播放到第 4 帧), 游戏过程中根据需要每个棋子影片剪辑播放不同帧。棋盘在设计时直接放在 Fla 文件的舞台上。

这里为了便于处理, 采用一个 qizi 二维数组用来存储棋盘上棋子状态, 一个 qipan 一维数组用来存储棋子影片。

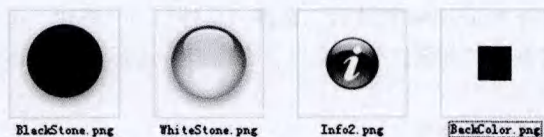


图 2 黑白两色棋子

2.2 翻转对方的棋子

需要从自己落子 (x1, y1) 为中心的横、竖、斜 8 个方向上判断是否需要翻转对方的棋子, 程序中由棋子影片剪辑 qi 的 MouseEvent.CLICK 事件实现的。在 MouseEvent.CLICK 事件中参数 event 对象含有被单击对象信息。event.target 可以获取被单击棋子影片剪辑对象 thisQi。

剪辑对象 thisQi 位置像素信息 (thisQi.x, thisQi.y) 转换成棋盘坐标 (x1, y1)。从左、左上、上、右上、右、右下、下、左下 8 个方向上调用过程 DirectReverse (x1, y1, dx, dy) 翻转对方的棋子。而具体棋子的翻转由 FanQi (x, y) 实现。FanQi (x, y) 修改数组 qizi 的 (x, y) 处保存棋盘上的棋子信息, 同时播放到指定帧。


```
private function FanQi(x, y:int):void {
    if (qizi[x][y] == BLACK) {
        qizi[x][y] = WHITE;
        qipan[8 * x + y].gotoAndStop(WHITE);
        //显示 WHITE 棋子图形
    } else {
        qizi[x][y] = BLACK;
        qipan[8 * x + y].gotoAndStop(BLACK);
        //显示 BLACK 棋子图形
    }
}
```

作者简介: 潘惠勇 (1977-), 男, 副教授, 硕士, 研究方向: 软件工程和云计算。




```
}
```

2.3 显示执棋方可落子位置

Can_go (x1, y1) 从左、左上、上、右上、右、右下、下、左下 8 个方向上调用函数 CheckDirect (x1, y1, dx, dy) 判断某方向上是否形成夹击之势, 如果形成且中间无空子则返回 True, 表示 (x1, y1) 可以落子。(x1, y1) 处可以落子则用  图片显示。

2.4 判断胜负功能

qizi 二维数组保存棋盘上的棋子信息, 其中元素保存 1, 表示此处为黑子; 元素保存 2, 表示此处为白子; 元素保存 0, 表示此处为无棋子。通过对 qizi 数组中各方棋子数统计, 在棋盘无处可下时, 根据各方棋子数判断出输赢。

3 设计步骤

3.1 创建 Flash 文件

打开 Flash CS6 软件后, 选择“文件”→“新建”选项, 系统将弹出“新建文档”窗口, 在窗口中选择“ActionScript 3.0”选项。

3.1.1 设置文档属性

选择菜单“修改”, 再选择“文档”选项, 调出“文档属性”对话框。设置场景的尺寸为 720*780 像素, 背景颜色为浅绿色, 然后单击“确定”按钮。在属性面板设置文档类为 Main。

3.1.2 设计棋子影片剪辑元件

选择菜单“插入”→“新建元件”。在新弹出的“新建元件”窗口中, 将元件名称设置为“棋子”, 将元件类型设置为“影片剪辑”, 单击“确定”按钮后, Flash 界面将转变为“棋子”元件的编辑区。导入图 2 所示 4 幅图, 注意每幅图大小 80*80。

3.2 设计游戏文档类 (Main.as)

选择“文件”→“新建”选项, 系统将弹出“新建文档”窗口。在窗口中选择“ActionScript 文件”选项。这样在 Flash 中新建一个 ActionScript 类文件, 将其命名为 Main.as。导入包及相关类:

```
package {
    import flash.display.*;
    import flash.events.*;
    import flash.text.*;
    import flash.utils.Timer;
```

类成员变量定义:

```
public class Main extends MovieClip {
    //常量
    private static const BLACK:int = 1;
    private static const WHITE:int = 2;
    private static const KONG:int = 0;
```

```
private var qizi:Array =new Array();//构造一个二维
//数组用来存储棋子状态
```

```
private var qipan:Array =new Array();//构造一个一
//维数组用来存储棋子影片
```



```
private var curQizi :int = BLACK;// 当前走棋方
var hitTimer:Timer=new Timer(1000);//计时器,定时
//清除提示图形
```

构造函数对保存棋盘上的棋子信息的 qizi 数组初始化, 实例化所有棋子影片对象。同时在棋盘上显示初始的 4 个棋子。

```
public function Main():void {
    //构造函数
    var i,j:int;
    for (i=0; i<8; i++) {
        qizi[i]=new Array();
        for (j=0; j<8; j++) {
            qizi[i][j]=KONG;
            var qi:Qi=new Qi();//棋子实例
            qi.y=80*j+42;//确定位置
            qi.x=80*i+42;
            qi.gotoAndStop(4);//显示棋子图形
            qipan.push(qi);
        }
        qi.addEventListener(MouseEvent.CLICK,clickQi);
        addChild(qi);//加到显示列表
    }
    // 棋盘上初始 4 个棋子
    qizi[3][3] = WHITE;
    qipan[8 * 3 + 3].gotoAndStop(2);//显示白色棋子图形
    qizi[4][4] = WHITE;
    qipan[8 * 4 + 4].gotoAndStop(2);//显示白色棋子图形
    qizi[3][4] = BLACK;
    qipan[8 * 3 + 4].gotoAndStop(1);//显示黑色棋子图形
    qizi[4][3] = BLACK;
    qipan[8 * 4 + 3].gotoAndStop(1);//显示黑色棋子图形
    help_btn.addEventListener (MouseEvent.
CLICK,clickHelp);
    message_txt.text="该黑棋走子";
}
```

构造函数同时对所有的棋子影片对象和“帮助”按钮添加鼠标单击事件的侦听。如果是 help_btn 按钮被单击, 则执行 clickHelp 函数显示可以落子的位置提示。

```
public function clickHelp(event:MouseEvent) {
    showCanPosition();//显示可以落子的位置
}
```

Show_Can_Position () 用图片  显示可以落子的位置。同时启动定时器控制图片  仅显示 1 秒钟。

```
private function showCanPosition():void {
    //显示可以落子的位置
    var i,j:int;
    var n:int = 0;//可以落子的位置统计
```


***** GAME PROGRAM *****

```

        for (i = 0; i <= 7; i++) {
            for (j = 0; j <= 7; j++) {
                if (qizi[i][j] == 0 && Can_go(i, j)) {
                    n = n + 1;
                    qipan[8 * i + j].gotoAndStop(3); //显示提示图形
                }
            }
        }
        hitTimer.start();
        hitTimer.addEventListener (TimerEvent.
TIMER, clsCanPosition);
    }
    private function clsCanPosition(event:Event) {
        var i,j:int;
        for (i = 0; i <= 7; i++) {
            for (j = 0; j <= 7; j++) {
                if (qizi[i][j] == 0 && Can_go(i, j)) {
                    qipan[8 * i + j].gotoAndStop(4); //显示背景图形
                }
            }
        }
        hitTimer.removeEventListener (TimerEvent.
TIMER, clsCanPosition);
    }
}

```

如果是棋子影片对象（此时显示是背景第4帧）被单击，则此剪辑对象 thisQi 位置像素信息（thisQi.x, thisQi.y）可以转换成棋盘坐标（x1, y1），然后判断当前位置（x1, y1）是否可以放棋子（符合夹角之势），如果可以则此棋子影片对象调用 gotoAndStop（curQizi）显示自己棋子图形，调用 FanALLQi（i, j）从左、左上、上、右上等8个方向翻转对方的棋。最后判断对方是否有棋可走，如果对方可以走棋则交换走棋方。如果对方不可以走棋，则自己可以继续走棋，直到双方都不能走棋，显示输赢信息。

```

public function clickQi(event:MouseEvent) {
    var x1:int, y1:int;
    var thisQi:Qi = (event.target as Qi); // what Qi?
    x1=(thisQi.x-42)/80;
    y1=(thisQi.y-42)/80;
    if (Can_go(x1, y1)) // 判断当前位置是否可以放棋子
        trace("can");
    qizi[x1][y1] = curQizi;
    FanALLQi(x1, y1); // 从8个方向翻转对方的棋
    qipan[8 * x1 + y1].gotoAndStop(curQizi); //显示棋子图形
    //统计棋盘已下棋子数量 n
    for (x = 0; x <= 7; x++) {
        for (y = 0; y <= 7; y++) {
            if (qizi[x][y] != 0)
                n = n + 1;
        }
    }
}

```

```

if(n==64) //棋盘已下满,显示输赢信息
    isLoseWin();
return;
}
//判断对方是否有棋可走,如有交换走棋方
if (curQizi==WHITE && checkNext(BLACK)
|| curQizi==BLACK && checkNext(WHITE)) {
    if (curQizi==WHITE) {
        curQizi=BLACK;
        message_txt.text="该黑棋走子";
    } else {
        curQizi=WHITE;
        message_txt.text="该白棋走子";
    }
} else if (checkNext(curQizi)) {
    //判断自己是否有棋可走,如有,给出提示
    message_txt.text="对方无棋可走,请继续";
} else //双方都无棋可走,游戏结束,显示输赢信息
    isLoseWin();
//统计双方的棋子数量,显示输赢信息。
} else {
    message_txt.text="不能落子! ";
}
}
}

```

Can_go (x1, y1) 从左、左上、上、右上、右、右下、下、左下8个方向判断（x1, y1）处可否落子。

```

private function Can_go(x1, y1:int):Boolean {
    //从左、左上、上、右上、右、右下、下、左下8个方
    //向判断
    if (CheckDirect(x1, y1, -1, 0) == true) return true;
    if (CheckDirect(x1, y1, -1, -1) == true) return true;
    if (CheckDirect(x1, y1, 0, -1) == true) return true;
    if (CheckDirect(x1, y1, 1, -1) == true) return true;
    if (CheckDirect(x1, y1, 1, 0) == true) return true;
    if (CheckDirect(x1, y1, 1, 1) == true) return true;
    if (CheckDirect(x1, y1, 0, 1) == true) return true;
    if (CheckDirect(x1, y1, -1, 1) == true) return true;
}

```

checkNext (i:int) 验证参数代表的走棋方是否还有棋可走。

```

/**
 * @param i 代表走棋方,1为黑方,2为白方
 * @return true/false
 */
private function checkNext(i:int):Boolean {
    if (Can_Num() > 0) {
        return true;
    } else {
        return false;
    }
}
}

```

Can_Num () 统计可以落子的位置数。




```
private function Can_Num():int //统计可以落子的
//位置数
var i, j, n = 0;
for (i = 0; i <= 7; i++) {
    for (j = 0; j <= 7; j++) {
        if (Can_go(i, j)) {
            n = n + 1;
        }
    }
}
return n; //可以落子的位置个数
}
```

isLoseWin () 统计双方的棋子数量，显示输赢信息。

```
// 显示输赢信息
private function isLoseWin() {
    var whitenum:int = 0;
    var blacknum:int = 0;
    var n = 0,x,y:int;
    for (x = 0; x <= 7; x++) {
        for (y = 0; y <= 7; y++) {
            if (qizi[x][y] != 0) {
                n = n + 1;
                if (qizi[x][y] == 2) {
                    whitenum += 1;
                }
                if (qizi[x][y] == 1) {
                    blacknum += 1;
                }
            }
        }
    }
    if (blacknum > whitenum) {
        message_txt.text = " 游戏结束黑方胜利,
黑方:" +
        String (blacknum) + " 白方:" + String
(whitenum);
    } else {
        message_txt.text = " 游戏结束白方胜利,
黑方:" +
        String (blacknum) + " 白方:" + String
(whitenum);
    }
}
```

FanALLQi (int x1, int y1) 从左、左上、上、右上、右、右下、下、左下 8 个方向翻转对方的棋子。

```
private function FanALLQi(x1, y1:int):void {
    //从左、左上、上、右上、右、右下、下、左下 8 个方
//向翻转
    if (CheckDirect(x1, y1, -1, 0) == true) {
        DirectReverse(x1, y1, -1, 0);
    }
```

```
if (CheckDirect(x1, y1, -1, -1) == true) {
    DirectReverse(x1, y1, -1, -1);
}
if (CheckDirect(x1, y1, 0, -1) == true) {
    DirectReverse(x1, y1, 0, -1);
}
if (CheckDirect(x1, y1, 1, -1) == true) {
    DirectReverse(x1, y1, 1, -1);
}
if (CheckDirect(x1, y1, 1, 0) == true) {
    DirectReverse(x1, y1, 1, 0);
}
if (CheckDirect(x1, y1, 1, 1) == true) {
    DirectReverse(x1, y1, 1, 1);
}
if (CheckDirect(x1, y1, 0, 1) == true) {
    DirectReverse(x1, y1, 0, 1);
}
if (CheckDirect(x1, y1, -1, 1) == true) {
    DirectReverse(x1, y1, -1, 1);
}
}
```

CheckDirect () 判断某方向上是否形成夹击之势，如果形成且中间无空子则返回 True。

```
private function CheckDirect (x1, y1, dx, dy:int):
Boolean {
    var x, y:int;
    var flag:Boolean;
    x = x1 + dx;
    y = y1 + dy;
    flag = false;
    while (InBoard(x, y) && !Ismychess(x, y) &&
qizi[x][y] != 0) {
        x += dx;
        y += dy;
        flag = true; //构成夹击之势
    }
    if (InBoard(x, y) && Ismychess(x, y) && flag == true) {
        return true; //该方向落子有效
    }
    return false;
}
```

DirectReverse () 针对已形成夹击之势某方向上的对方棋子进行翻转。

```
private function DirectReverse (x1, y1, dx, dy:int):
void {
    var x, y:int;
    var flag:Boolean;
    x = x1 + dx;
    y = y1 + dy;
    flag = false;
```



GAME PROGRAM

```

while (InBoard(x, y) && ! Ismychess(x, y) &&
qizi[x][y] != 0) {
    x += dx;
    y += dy;
    flag = true; //构成夹击之势
}
if (InBoard(x, y) && Ismychess(x, y) && flag
== true) {
    do {
        x -= dx;
        y -= dy;
        if ((x != x1 || y != y1)) {
            FanQi(x, y);
        }
    } while ((x != x1 || y != y1));
}
}

```

FanQi (int x, int y) 将存储 (x, y) 处棋子信息 qizi [x] [y] 的反色处理。

```

private function FanQi(x, y:int):void {
    if (qizi[x][y] == BLACK) {
        qizi[x][y] = WHITE;
        qipan[8 * x + y].gotoAndStop(WHITE);
    }
}
//显示棋子图形

```

```

} else {
    qizi[x][y] = BLACK;
    qipan[8 * x + y].gotoAndStop(BLACK);
}
//显示棋子图形
}

```

InBoard () 判断 (x,y) 是否在棋盘界内, 如果在界内则返回真, 否则返回假。

```

private function InBoard(x,y :int):Boolean {
    if (x >= 0 && x <= 7 && y >= 0 && y <= 7) {
        return true;
    } else {
        return false;
    }
}

```

至此就完成黑白棋游戏设计了, 运行程序效果如图 1 所示。

4 结语

用 Flash ActionScript3.0 实现经典的黑白棋游戏基本功能, 并且能够判断输赢, 如果用户可以进一步改进翻转效果和增加双方棋子数量提示, 则使得游戏更具吸引力。

(收稿日期: 2014-01-05)

知识产权结新果, 百尺竿头更进步

——飞天诚信韩雪峰副总受邀出席第三届北京市发明专利奖颁奖大会并接受国家知识产权局局长申长雨先生的颁奖

近日, 北京市知识产权局在京举行第三届北京市发明专利奖颁奖大会, 隆重表彰对积极运用知识产权制度, 推动技术创新和促进首都经济社会发展做出突出贡献的发明人、和专利权人。北京市市长王安顺、副市长戴均良、国家知识产权局局长申长雨、北京市知识产权局汪洪局长、周砚副局长出席会议并讲话, 同时也为获得第三届北京市发明专利奖的企业代表颁奖。国家知识产权局局长申长雨先生亲自为飞天诚信副总经理韩雪峰颁奖。

据悉, 北京市发明专利奖是全国首个省部级的发明专利奖, 自 2008 年起, 每两年举行一届, 主要用来表彰为本市经济社会发展作出突出贡献的专利权人和发明人, 鼓励先进技术的持有人将其创造成果申请专利, 以法律的形式将其权利加以固定。第三届北京市发明专利奖从 8 个技术领域 182 个申报项目中评选出获奖项目 36 项。包含清华大学、联想集团、北大方正、腾讯、飞天诚信等在内的 36 家优秀企业获此殊荣。

飞天诚信经过历时 9 个月激烈角逐最终凭借“通用串

行总线数据传输方法”的发明专利从此次参选的众多项目中脱颖而出, 荣获三等奖, 是信息安全领域唯一获奖企业, 也是历届评选中唯一一家获奖的信息安全领域企业。在申长雨局长的颁奖期间, 韩雪峰副总简单介绍了公司的情况, 申局长对飞天诚信获奖表示恭喜和祝贺, 希望公司能珍惜荣誉, 再接再厉, 为建设知识产权首善之区做出新的更大的贡献。

优秀的专利项目是城市的科技名片, 作为以自主研发为主的国家高新技术软件企业, 飞天诚信始终视创新为企业立命之本, 视知识产权为企业核心竞争力的重要组成部分。经过十余年的不懈努力, 飞天已形成完善的知识产权保护体系。截止目前, 飞天诚信已申请专利近千项, 已授权专利 600 余项。其中专利发明授权 400 余项, 企业每百人发明专利拥有量近 65 件。是同行业内拥有国内及国外专利申请及授权量领先的企业。公司先后被评为北京市专利试点先进单位、北京市专利示范企业、全国企事业知识产权优势单位等多项荣誉。

