19.2.4 设计推箱子游戏

当用户操作键盘时会触发键盘事件，键盘事件主要包括下面 3 种类型：

- keydown：在键盘上按下某个键时触发。如果按住某个键，会不断触发该事件，但是 Opera 浏览器不支持这种连续操作。该事件处理函数返回 false 时，会取消默认的动作（如输入的键盘字符，在 IE 和 Safari 浏览器下还会禁止 keypress 事件响应）。
- keypress：按下某个键盘键并释放时触发。如果按住某个键，会不断触发该事件。该事件处理函数返回 false 时，会取消默认的动作（如输入的键盘字符）。
- keyup：释放某个键盘键时触发。该事件仅在松开键盘时触发一次，不是一个持续的响应状态。

键盘事件定义了很多属性，利用这些属性可以精确控制键盘操作，说明如表 E19.2 所示。

表 E19.2 键盘事件定义的属性

属性	说明
keyCode	该属性包含键盘中对应键位的键值
charCode	该属性包含键盘中对应键位的 Unicode 编码，仅 DOM 支持
target	发生事件的节点（包含元素），仅 DOM 支持
srcElement	发生事件的元素，仅 IE 支持
shiftKey	是否按下 Shift 键，如果按下返回 true，否则为 false
ctrlKey	是否按下 Ctrl 键，如果按下返回 true，否则为 false
altKey	是否按下 Alt 键，如果按下返回 true，否则为 false
metaKey	是否按下 Meta 键，如果按下返回 true，否则为 false，仅 DOM 支持

键盘事件属性一般只在键盘相关事件发生时才会存在于事件对象中，但是 ctrlKey 和 shiftKey 属性除外，因为它们可以在鼠标事件中存在，如当按下 Ctrl 键或 Shift 键时单击鼠标操作。

keyCode 和 charCode 属性使用比较复杂，但是它们在实际开发中又比较常用，故比较这两个属性在不同事件类型和不同浏览器中的表现是非常必要的，详细说明如表 E19.3 所示。

表 E19.3 keyCode 和 charCode 属性值

属性（响应事件类型）	IE 事件模型	标准事件模型
keyCode（keypress）	返回所有字符键的正确值，区分大写状态（65-90）和小写状态（97~122）	功能键返回正确值，而【Shift】、【Ctrl】、【Alt】、【PrintScreen】、【ScrollLock】无返回值，其他所有键值都返回 0
keyCode（keydown）	返回所有键值（除【PrintScreen】键），字母键都以大写状态显示键值（65~90）	返回所有键值（除【PrintScreen】键），字母键都以大写状态显示键值（65-90）
keyCode（keyup）	返回所有键值（除【PrintScreen】键），字母键都以大写状态显示键值（65~90）	返回所有键值（除【PrintScreen】键），字母键都以大写状态显示键值（65-90）
charCode（keypress）	不支持该属性	返回字符键，区分大写状态（65-90）和小写状态（97-122），【Shift】、【Ctrl】、【Alt】、【PrintScreen】、【ScrollLock】键无返回值，其他所有键值为 0
charCode（keydown）	不支持该属性	所有键值为 0
charCode（keyup）	不支持该属性	所有键值为 0

某些键的可用性不是很确定，如 PageUp 键和 Home 键等。不过常用功能键和字符键都是比较稳定的，如表 E19.4 所示。

表 E19.4 键位和码值对照表

键位	码值	键位	码值
0~9（数字键）	48~57	A~Z（字母键）	65~90
Backspace（退格键）	8	Tab（制表键）	9
Enter（回车键）	13	Space（空格键）	32

Left arrow（左箭头键）	37	Top arrow（上箭头键）	38
Right arrow（右箭头键）	39	Down arrow（下箭头键）	40

下面我们来设计推箱子游戏的动画效果，关于游戏本身的算法、策划和逻辑设计不是本示例要讲解的重点。

【操作步骤】

第 1 步，启动 Dreamweaver，新建文档，保存为 test.html。设计一个简单的文档结构：一个 标签，用来设计游戏背景；一个被推动的箱子 。

```
<body>
<div id="wrap">
    <div id="box"><img src="images/box.png" /></div>
</div>
</body>
```

第 2 步，在头部标签<head>内插入一个<style type="text/css">标签，设计内部样式表。为 标签标签定义游戏背景，设计相对定位，为推动的箱子确定定位包含框；为 标签定义大小、定位。页面显示效果如图 E19.3 所示，详细代码如下所示：

```
<style type="text/css">
#wrap {/* 定义游戏背景，设计定位包含框 */
    background:url(images/bg.png) no-repeat;
    border:solid 1px red;
    width:714px;
    height:539px;
    position:relative;
/* 定义定位包含框，推动的箱子在该框中定位 */
}
#box {/* 定义推动的箱子的大小、定位 */
    width:35px;                    /* 箱子宽 */
    height:35px;                   /* 箱子高 */
    position:absolute;             /* 绝对定位 */
    left:72px;                     /* 默认显示在第一个格子中 */
    top:26px;                      /* 默认显示在第一个格子中 */
}
</style>
```

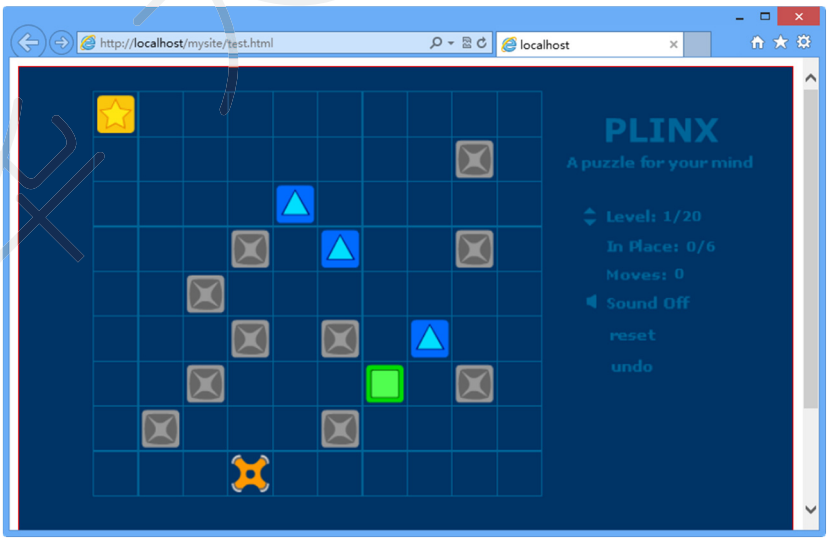

图 E19.3 页面初始化设计效果

第 3 步，设计 Javascript 脚本：设计一个驱动函数 keyDown()，该函数能够获取键盘被按下的键，然后进行判断：如果是左箭头键，则向左推动箱子 42 个像素；如果是右箭头键，则向右推动箱子 42 个像素；如果是上箭头键，则向上推动箱子 42 个像素；如果是下箭头键，则向下推动箱子 42 个像素。

第 4 步，在页面加载完毕之后调用 load 事件处理函数，获取箱子对象，在箱子对象上绑 keydown 事件，把 keyDown()函数传递给事件句柄，这样每当按下键盘键时都将实施调用驱动函数 keyDown()，根据侦测的键值，判断是否推动以及如何推动箱子。完整脚本如下所示。

```
<script>
function keyDown(event){
// 方向键控制元素移动函数
    var event = event || window.event;
// 标准化事件对象
    switch(event.keyCode){
// 获取当前按下键盘键的编码
        case 37 :
// 按下左箭头键，向左移动 42 个像素
            box.style.left = box.offsetLeft - 42   + "px";
            break;
        case 39 :
// 按下右箭头键，向右移动 42 个像素
            box.style.left = box.offsetLeft + 42 + "px";
            break;
        case 38 :
// 按下上箭头键，向上移动 42 个像素
            box.style.top = box.offsetTop   - 42 + "px";
            break;
        case 40 :
// 按下下箭头键，向下移动 42 个像素
            box.style.top = box.offsetTop   + 42 + "px";
            break;
    }
    return false
}
window.onload = function(){
            var box = document.getElementById("box");
// 获取页面元素的引用指针
            box.style.position = "absolute";
// 色块绝对定位
            document.onkeydown = keyDown;
// 在 Document 对象中注册 keyDown 事件处理函数
}
</script>
```

第 5 步，在浏览器中预览，然后在键盘上分别按动方向键，可以看到箱子在方格中自由推动跳转，如图 E19.4 所示。

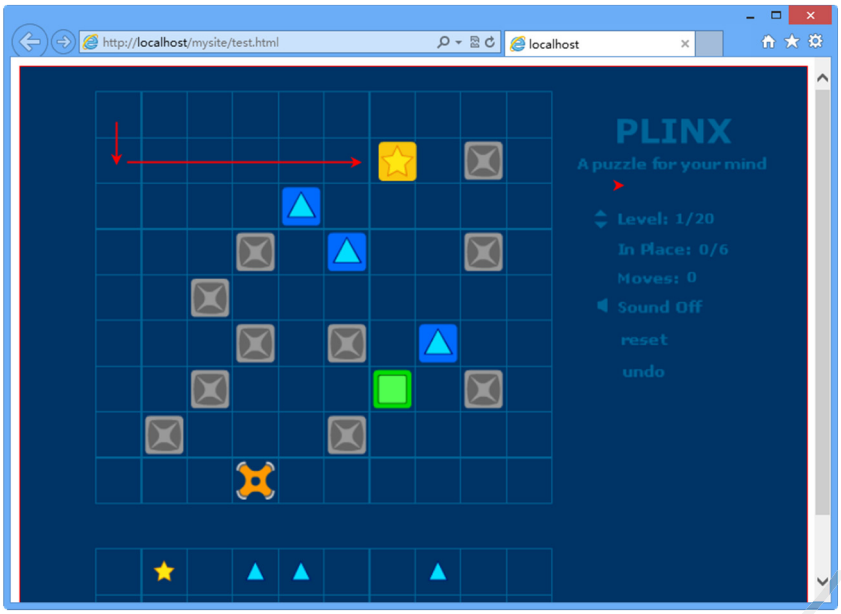

图 E19.4　使用键盘推动箱子效果