2024年夏季《移动软件开发》实验报告

姓名：刘瑜 学号：22020006076

姓名和学号?	**刘瑜，22020006076**
本实验属于哪门课程?	中国海洋大学24夏《移动软件开发》
实验名称?	实验6：推箱子游戏
博客地址?	2024年《移动软件开发》实验六：推箱子游戏-CSDN博客
Github仓库地址?	Yulia2233/mobile_software_project: 移动软件开发，实验六 (github.com)

（备注：将实验报告发布在博客、代码公开至 github 是 **加分项**，不是必须做的）

（备注：PDF无法播放动图，效果展示的动图放在了CSDN的文章中）

一、实验目标

1、综合所学知识创建完整的推箱子游戏；2、能够在开发过程中熟练掌握真机预览、调试等操作。

（一）基本功能

1.包含四个关卡，列表中可以预览关卡内容

2.点开关卡是相应的游戏画面

3.正常的游戏功能，包括移动、重新开始、游玩成功

（二）额外功能

1.提供**撤回**功能，点击撤回按钮可以撤回到上一步

2.提供**保存**功能。中途退出游戏后，再次点击，可以恢复到上次打开的进度。

二、实验步骤

1.首页设计

```
<!--pages/menu/menu.wxml-->
<view class='container'>
  <!-- 标题 -->
  <view class='title'>游戏选关</view>
  <!-- 关卡列表 -->
  <view class='levelBox'>
```

```
    <view class='box' wx:for='{{levels}}' wx:key='index' bindtap='chooseLevel'
data-level='{{index}}'>
      <image src='/images/{{item}}'></image>
      <text>第{{index+1}}关</text>
    </view>
  </view>
</view>
```

```
/* 关卡列表区域 */
.levelBox {
  width: 100%;
}

/* 单个关卡区域 */
.box {
  width: 50%;
  float: left;
  margin: 20rpx 0;
  display: flex;
  flex-direction: column;
  align-items: center;
}

/* 选关图片 */
image {
  width: 300rpx;
  height: 300rpx;
}

.title {
  font-size: 20px;
}
```

2.公共逻辑

```
//关卡1
var map1 = [
  [0, 1, 1, 1, 1, 1, 0, 0],
  [0, 1, 2, 2, 1, 1, 1, 0],
  [0, 1, 5, 4, 2, 2, 1, 0],
  [1, 1, 1, 2, 1, 2, 1, 1],
  [1, 3, 1, 2, 1, 2, 2, 1],
  [1, 3, 4, 2, 2, 1, 2, 1],
  [1, 3, 2, 2, 2, 4, 2, 1],
  [1, 1, 1, 1, 1, 1, 1, 1]
]
//关卡2
var map2 = [
  [0, 0, 1, 1, 1, 0, 0, 0],
  [0, 0, 1, 3, 1, 0, 0, 0],
  [0, 0, 1, 2, 1, 1, 1, 1],
  [1, 1, 1, 4, 2, 4, 3, 1],
  [1, 3, 2, 4, 5, 1, 1, 1],
  [1, 1, 1, 1, 4, 1, 0, 0],
```

```
  [0, 0, 0, 1, 3, 1, 0, 0],
  [0, 0, 0, 1, 1, 1, 0, 0]

]
//关卡3
var map3 = [
  [0, 0, 1, 1, 1, 1, 0, 0],
  [0, 0, 1, 3, 3, 1, 0, 0],
  [0, 1, 1, 2, 3, 1, 1, 0],
  [0, 1, 2, 2, 4, 3, 1, 0],
  [1, 1, 2, 2, 5, 4, 1, 1],
  [1, 2, 2, 1, 4, 4, 2, 1],
  [1, 2, 2, 2, 2, 2, 2, 1],
  [1, 1, 1, 1, 1, 1, 1, 1]
]
//关卡4
var map4 = [
  [0, 1, 1, 1, 1, 1, 1, 0],
  [0, 1, 3, 2, 3, 3, 1, 0],
  [0, 1, 3, 2, 4, 3, 1, 0],
  [1, 1, 1, 2, 2, 4, 1, 1],
  [1, 2, 4, 2, 2, 4, 2, 1],
  [1, 2, 1, 4, 1, 1, 2, 1],
  [1, 2, 2, 2, 5, 2, 2, 1],
  [1, 1, 1, 1, 1, 1, 1, 1]
]

module.exports = {
  maps: [map1, map2, map3, map4]
}
```

3.游戏逻辑设计

游戏页面如下：

```
  <!-- 游戏画布 -->
  <canvas canvas-id='myCanvas'></canvas>

  <!-- 方向键 -->
  <view class='btnBox'>
    <button type='warn' bindtap='up'>↑</button>
    <view>
      <button type='warn' bindtap='left'>←</button>
      <button type='warn' bindtap='down'>↓</button>
      <button type='warn' bindtap='right'>→</button>
    </view>
  </view>
  <button  bindtap="restoreState">撤回</button>
  <!-- 重新开始按钮 -->
  <button type='warn' bindtap='restartGame'>重新开始</button>
</view>
```

地图展示逻辑：

```
initMap: function(level) {
    // 读取原始的游戏地图数据
    let mapData = data.maps[level]
    //使用双重for循环记录地图数据
    for (var i = 0; i < 8; i++) {
      for (var j = 0; j < 8; j++) {
        box[i][j] = 0
        map[i][j] = mapData[i][j]

        if (mapData[i][j] == 4) {
          box[i][j] = 4
          map[i][j] = 2
        } else if (mapData[i][j] == 5) {
          map[i][j] = 2
          //记录小鸟的当前行和列
          row = i
          col = j
        }
      }
    }
  },
  /**
   * 自定义函数--绘制地图
   */
  drawCanvas: function() {
    let ctx = this.ctx
    //清空画布
    ctx.clearRect(0, 0, 320, 320)
    //使用双重for循环绘制8x8的地图
    for (var i = 0; i < 8; i++) {
      for (var j = 0; j < 8; j++) {
        //默认是道路
        let img = 'ice'
        if (map[i][j] == 1) {
          img = 'stone'
        } else if (map[i][j] == 3) {
          img = 'pig'
        }

        //绘制地图
        ctx.drawImage('/images/icons/' + img + '.png', j * w, i * w, w, w)

        if (box[i][j] == 4) {
          //叠加绘制箱子
          ctx.drawImage('/images/icons/box.png', j * w, i * w, w, w)
        }
      }
    }

    //叠加绘制小鸟
    ctx.drawImage('/images/icons/bird.png', col * w, row * w, w, w)

    ctx.draw()
  },
```

移动逻辑：

```
up: function() {
    this.saveState();
    //如果不在最顶端才考虑上移
    if (row > 0) {
      //如果上方不是墙或箱子,可以移动小鸟
      if (map[row - 1][col] != 1 && box[row - 1][col] != 4) {
        //更新当前小鸟坐标
        row = row - 1
      }
      //如果上方是箱子
      else if (box[row - 1][col] == 4) {
        //如果箱子不在最顶端才能考虑推动
        if (row - 1 > 0) {
          //如果箱子上方不是墙或箱子
          if (map[row - 2][col] != 1 && box[row - 2][col] != 4) {
            box[row - 2][col] = 4
            box[row - 1][col] = 0
            //更新当前小鸟坐标
            row = row - 1
          }
        }
      }
      //重新绘制地图
      this.drawCanvas()
      //检查游戏是否成功
      this.checkWin()
    }
  },
  /**
   * 自定义函数--方向键：下
   */
  down: function() {
    this.saveState();
    //如果不在最底端才考虑下移
    if (row < 7) {
      //如果下方不是墙或箱子,可以移动小鸟
      if (map[row + 1][col] != 1 && box[row + 1][col] != 4) {
        //更新当前小鸟坐标
        row = row + 1
      }
      //如果下方是箱子
      else if (box[row + 1][col] == 4) {
        //如果箱子不在最底端才能考虑推动
        if (row + 1 < 7) {
          //如果箱子下方不是墙或箱子
          if (map[row + 2][col] != 1 && box[row + 2][col] != 4) {
            box[row + 2][col] = 4
            box[row + 1][col] = 0
            //更新当前小鸟坐标
            row = row + 1
          }
        }
      }
```

```
      //重新绘制地图
      this.drawCanvas()
      //检查游戏是否成功
      this.checkWin()
    }
  },
  /**
   * 自定义函数--方向键：左
   */
  left: function() {
    this.saveState();
    //如果不在最左侧才考虑左移
    if (col > 0) {
      //如果左侧不是墙或箱子,可以移动小鸟
      if (map[row][col - 1] != 1 && box[row][col - 1] != 4) {
        //更新当前小鸟坐标
        col = col - 1
      }
      //如果左侧是箱子
      else if (box[row][col - 1] == 4) {
        //如果箱子不在最左侧才能考虑推动
        if (col - 1 > 0) {
          //如果箱子左侧不是墙或箱子
          if (map[row][col - 2] != 1 && box[row][col - 2] != 4) {
            box[row][col - 2] = 4
            box[row][col - 1] = 0
            //更新当前小鸟坐标
            col = col - 1
          }
        }
      }
      //重新绘制地图
      this.drawCanvas()
      //检查游戏是否成功
      this.checkWin()
    }


  },
  /**
   * 自定义函数--方向键：右
   */
  right: function() {
    this.saveState();
    //如果不在最右侧才考虑右移
    if (col < 7) {
      //如果右侧不是墙或箱子,可以移动小鸟
      if (map[row][col + 1] != 1 && box[row][col + 1] != 4) {
        //更新当前小鸟坐标
        col = col + 1
      }
      //如果右侧是箱子
      else if (box[row][col + 1] == 4) {
        //如果箱子不在最右侧才能考虑推动
        if (col + 1 < 7) {
          //如果箱子右侧不是墙或箱子
```

```
          if (map[row][col + 2] != 1 && box[row][col + 2] != 4) {
            box[row][col + 2] = 4
            box[row][col + 1] = 0
            //更新当前小鸟坐标
            col = col + 1
          }
        }
      }
      //重新绘制地图
      this.drawCanvas()
      //检查游戏是否成功
      this.checkWin()
    }
  },
```

判断胜利逻辑：

```
  iswin: function() {
    //使用双重for循环遍历整个数组
    for (var i = 0; i < 8; i++) {
      for (var j = 0; j < 8; j++) {
        //如果有箱子没在终点
        if (box[i][j] == 4 && map[i][j] != 3) {
          //返回假，游戏尚未成功
          return false
        }
      }
    }
    //返回真，游戏成功
    return true
  },

  /**
   * 自定义函数--游戏成功处理
   */
  checkWin: function() {
    if (this.iswin()) {
      wx.showModal({
        title: '恭喜',
        content: '游戏成功! ',
        showCancel: false
      })
    }
  },
```

###

4.游戏额外逻辑

增加histort，存储游戏进度：

```
data: {
  level: 1,
  history: []  // 用于存储游戏状态的历史记录
},
```

将当前一步压入栈中：

```
saveState: function() {
  // 将当前的地图状态、箱子状态和小鸟的位置保存到history栈中
  this.data.history.push({
    map: JSON.parse(JSON.stringify(map)),
    box: JSON.parse(JSON.stringify(box)),
    row: row,
    col: col
  });
},
```

撤回当前一步：

```
restoreState: function() {
  if (this.data.history.length > 0) {
    let lastState = this.data.history.pop();
    map = JSON.parse(JSON.stringify(lastState.map));
    box = JSON.parse(JSON.stringify(lastState.box));
    row = lastState.row;
    col = lastState.col;
    this.drawCanvas();
  }
},
```

关掉游戏时，保存游戏进度：

```
onUnload: function() {
  wx.setStorageSync('currentMap', map);
  wx.setStorageSync('currentBox', box);
  wx.setStorageSync('currentRow', row);
  wx.setStorageSync('currentCol', col);
  wx.setStorageSync('history', this.data.history);
}
```

三、程序运行结果

1.撤回功能

推箱子游戏

第2关

↑

← ↓ →

撤回

重新开始

推箱子游戏

第2关

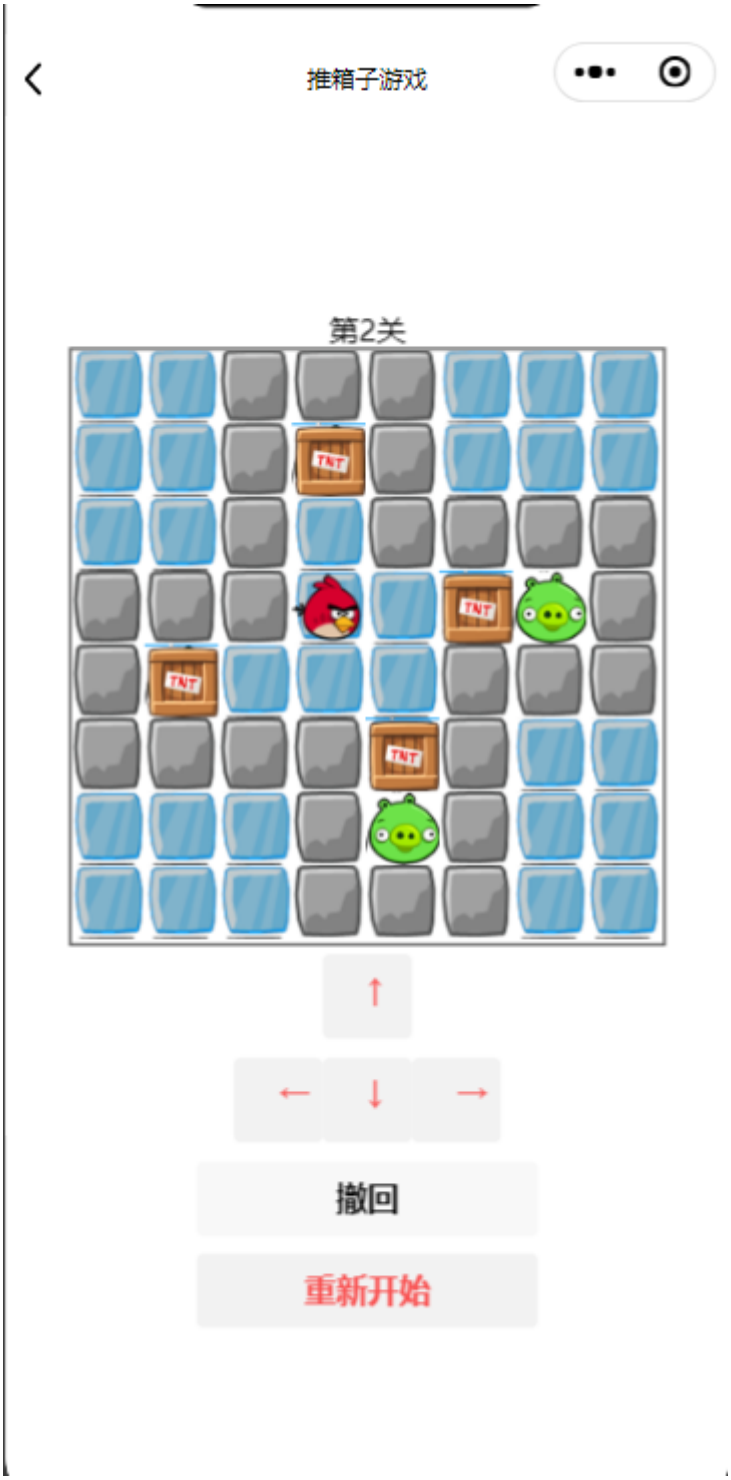
↑

← ↓ →

撤回

重新开始

2.保存游戏进度功能



退出重进后:

15:59 100%

推箱子游戏

第1关

↑

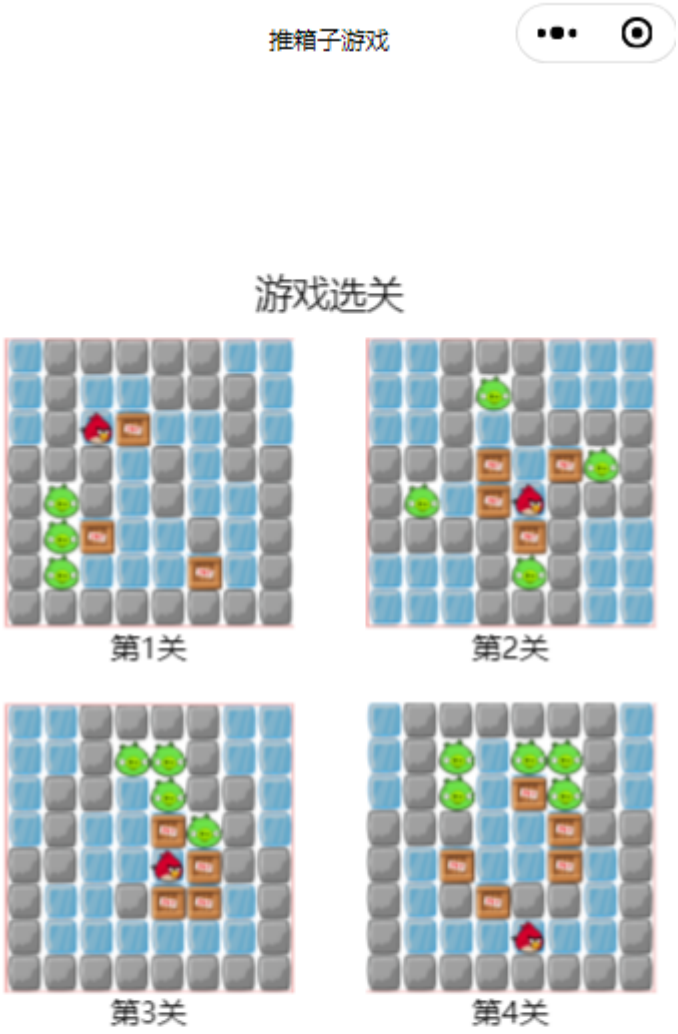
← ↓ →

撤回

重新开始

3.移动展示和游戏成功

4.主页展示



四、问题总结与体会

描述实验过程中所遇到的问题，以及是如何解决的。有哪些收获和体会，对于课程的安排有哪些建议。

问题：JavaScript 中对象和数组是按引用传递的。如果直接将 `map` 和 `box` 保存到 `history` 栈中，由于数组是引用类型，任何对 `map` 和 `box` 的更改都会影响到存储在 `history` 栈中的值，导致无法正确回滚状态。

解决方案：在保存状态时，使用 `JSON.stringify` 和 `JSON.parse` 深拷贝数据，这样可以避免引用问题。

```
saveState: function() {
  this.data.history.push({
    map: JSON.parse(JSON.stringify(map)),
    box: JSON.parse(JSON.stringify(box)),
    row: row,
    col: col
  });
}
```

收获和体会：编写这个推箱子游戏让我深刻体会到模块化编程和数据结构的重要性。通过合理组织代码和使用栈来实现撤回功能，我学会了如何高效管理游戏状态。同时，处理用户行为与状态保存的同步问题让我更深入理解了小程序的本地存储机制。这次开发不仅提升了我的编程技巧，也增强了对用户体验优化的理解。