2025年夏季《移动软件开发》实验报告

姓名：邓林 学号：23020007014

姓名和学号	**邓林，23020007014**
本实验属于哪门课程?	中国海洋大学25夏《移动软件开发》
实验名称?	实验6：推箱子游戏
博客地址?	2025年夏季《移动软件开发》实验报告6-CSDN博客
Github仓库地址?	https://github.com/xixiyhaha/2025Mobile-software-development.git

（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）

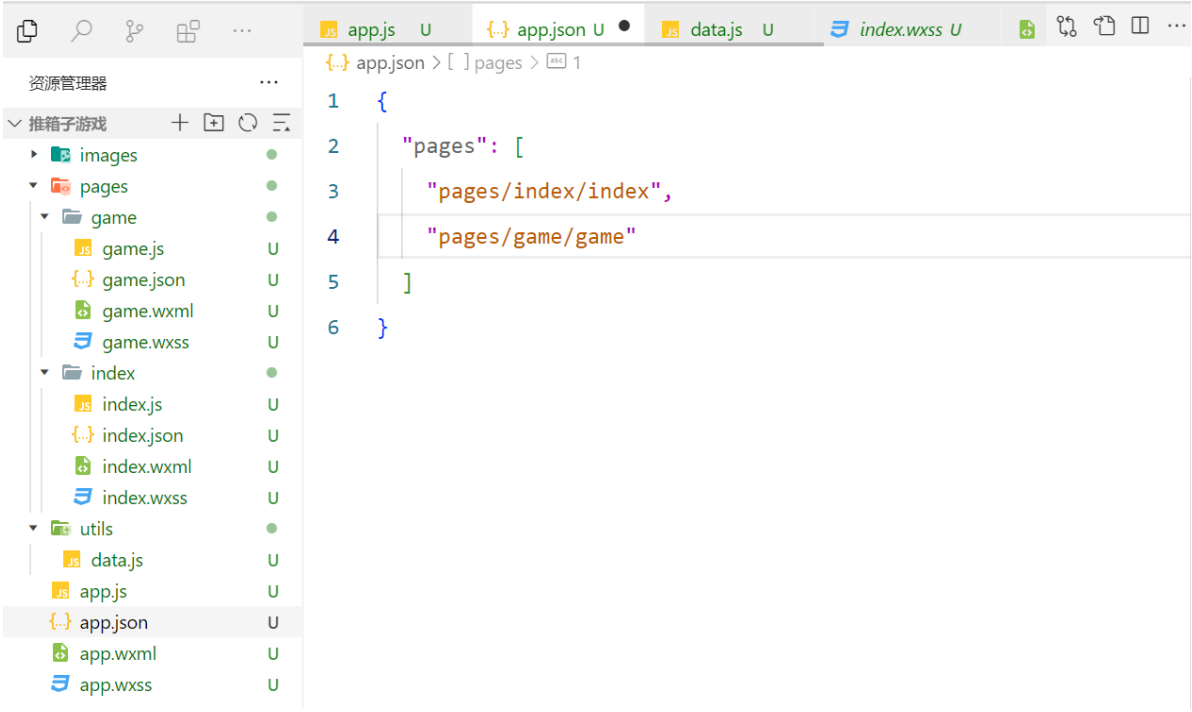
一、实验目标

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。

二、实验步骤

1、创建新文件，删除文件所有内容并按下图创建文件：



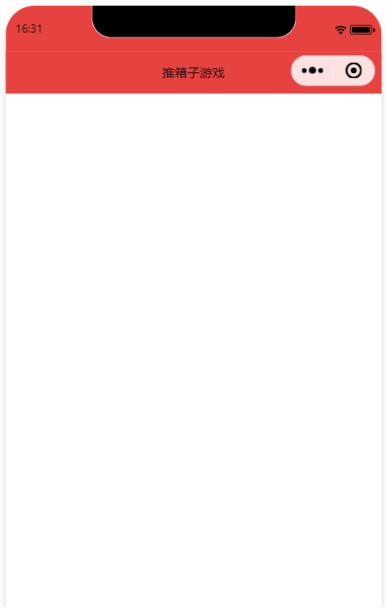
2、将图片素材导入images文件夹，链接：https://gaopursuit.oss-cn-beijing.aliyuncs.com/course/mobileDev/boxgame_images.zip；

3、配置导航栏样式：

app.js U | app.json U | data.js U | index.wxss U

app.json > ...

```
{
  "pages": [
    "pages/index/index",
    "pages/game/game"
  ],
  "window": {
    "navigationBarBackgroundColor": "#E64340",
    "navigationBarTextStyle": "black",
    "navigationBarTitleText": "推箱子游戏",
    "backgroundColor": "#eeeeee",
    "backgroundTextStyle": "light",
    "enablePullDownRefresh": true
  }
}
```



4、公共样式设计：

```
/* app.wsxx */
.container{
 height: 100vh;
 color: #E64340;
 font-weight:bold;
 display:flex;
 flex-direction:column;
 align-items:center;
 justify-content:space-evenly;
}

.title{
 font-size:18pt;
}
```

5、首页界面设计，样式如下：

```
<!--pages/index/index.wxml-->
<view class='container'>
  <!-- 标题 -->
  <view class='title'>游戏选关</view>
  <!-- 关卡列表 -->
    <view class='levelBox'>
    <view class='box'>
      <image src='/images/level01.png'></image>
      <text>第1关</text>
    </view>
    </view>
</view>
```

```
/* pages/index/index.wxss */
```

```
.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;
}
```

6、游戏页面设置：由于还没有制作进入游戏页面的入口，暂时没有做点击跳转的逻辑设计,所以可以在开发工具顶端选择“普通编译”下的“添加编译模式”,并携带临时测试参数 level=0。此时预览就可以直接显示game页面了,设计完毕后再改回“普通编译”模式即可重新显示首页；

7、游戏页面样式设计：

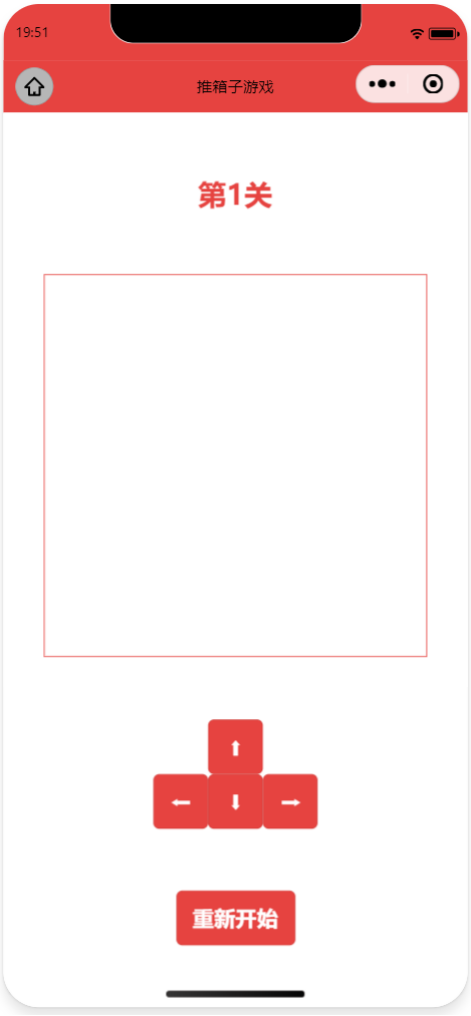
```
<!--pages/game/game.wxml-->
<view class='container'>
<!--关卡提示-->
<view class='title'>第1关</view>
<!--游戏画布-->

<canvas canvas-id='myCanvas'></canvas >

<!--方向键-->
<view class='btnBox'>
  <button type = 'warn'>⬆</button >
  <view>
    <button type= 'warn'>⬅</button >
    <button type= 'warn'>⬇</button >
    <button type='warn'>➡</button >
    </view>
  </view>
  <!--“重新开始”按钮-->
  <button type='warn'>重新开始</button>
</view>
```

```
/* pages/game/game.wxss */
/*游戏画布样式*/
Canvas{
  border:1rpx solid;
  width:320px;
  height:320px;
}
/*方向键按钮整体区域*/
.btnBox{
  display:flex;
  flex-direction:column;
  align-items:center;
}
/*方向键按钮第二行*/
.btnBox view{
```

```
  display:flex;
  flex-direction:row;
}
/*所有方向键按钮*/
.btnBox button{
  width:90rpx;
  height:90rpx;
}
/*所有按钮样式*/
.button{
  margin:10rpx;
}
```



8、在公共JS文件(utils/data.js)中配置游戏地图的数据,代码如下:

```
//地图数据map1~map4
//地图数据:1为墙、2为路、3为终点、4为箱子、5为人物、0为墙的外围

//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]
]

//暴露数据出口
module.exports={
  maps:[map1,map2,map3,map4]
}
```

9、公共逻辑处理：在game页的 JS 文件顶端引用公共 JS 文件：

```
// pages/game/game.js
var data = require('../../utils/data')//只能相对路径

Page({
......
```

10、关卡列表展示：在JS文件的data中录入关卡图片的数据信息，这里以4个关卡为例。相关JS (pages/index/index.js)代码片段如下：

```
Page({
  /**
   * 页面的初始数据
   */
  data: {
      levels:[
        'level01. png',
        'level02.png',
        'level03.png',
        'level04.png'
      ]
  },
```

11、接着为关卡对应的组件添加 wx:for 属性循环显示关卡列表数据和图片修改后的 WXML (pages/index/index.wxml) 代码如下:

```
<!--pages/index/index.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='levels{{index}}'
bindtap='chooseLevel' data-level='{{index}}'>
      < image src ='/images/{{item}}'></image>
      <text>第{{index + 1}}关</text>
    </view>
  </view>
</view >
```

12、上述代码表示为关卡添加了自定义点击事件函数chooseLevel，并且使用 data-level 属性携带了关卡图片下标信息。在对应的index.js文件中添加chooseLevel函数的内容,代码片段如下:

```
Page({
  // 自定义函数--游戏选关
  chooseLevel:function(e){
    let level = e.currentTarget.dataset.level
    wx.navigateTo({
      url:'../game/game?level='+ level
    })
  },
})
```

13、在首页逻辑中已经实现了页面跳转并携带了关卡对应的图片信息,现在需要在游戏页面接收关卡信息,并显示对应的图片内容;

相关 JS (pages/game/game.js) 代码片段如下:

```
Page({
  /**
   * 页面的初始数据
   */
  data: {
    level:1
  },
  /**
   * 生命周期函数--监听页面加载
   */
  onLoad:function(options){
    //获取关卡
    let level= options.level
    //更新页面关卡标题
    this.setData({
      level:parseInt(level)+ 1
    })
  },
)}
```

修改WXML(pages/game/game.wxml)代码片段如下:

```
<view class='container'>
<!--关卡提示-->
<view class='title'>第{{level}}关</view>
......
</view>
```

这样重新从首页点击不同的关卡图片跳转就可以发现已经能够正确显示对应的内容了；

14、游戏逻辑运行实现：首先在game.js文件的顶端记录一些游初始数据信息，对应的 JS (pages/game/game.js)代码段添加如下：

```
//地图图层数据
var map =[
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0]
]
//箱子图层数据
var box=[
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0],
  [0,0,0,0,0,0,0,0]
]

//方块的宽度
var w= 40
//初始化游戏主角(小鸟)的行与列
var row = 0
var col = 0
```

15、初始化游戏画面：根据当前是第几关读取对应的游戏地图信息,并更新到游戏初始数据中。在 game.js 文件中添加 initMap 函数,用于初始化游戏地图数据对应的 JS (pages/game/game.js)代码片段添加如下:

```
/**
   * 自定义函数--初始化地图数据
   */
  initMap:function(level){
    //读取原始的游戏地图数据
    let mapData = data.maps[level]
    //使用双重 for 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
        }
      }
    }
  },
```

上述代码首先从公共函数文件 data.js 中读取对应关卡的游戏地图数据，然后使用双重for 循环对每一块地图数据进行解析，并更新当前游戏的初始地图数据、箱子数据以及游戏主角(小鸟)的所在位置。

16、在 game.js 中添加自定义两数 drawCanvas，用于将地图信息绘制到画布上。对应的 JS (pages/game/game.js)代码片段添加如下:

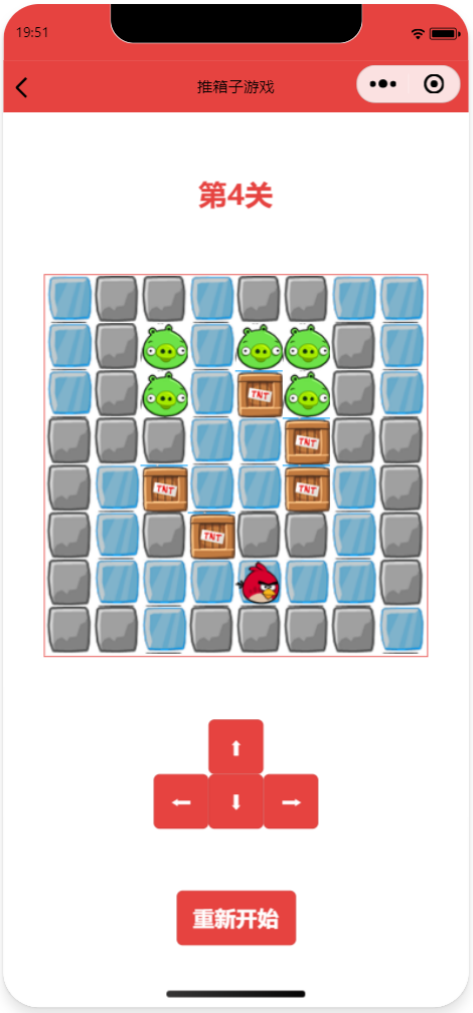
```
/**
   * 自定义函数--绘制地图
  */
  drawCanvas:function(){
    let ctx = this.ctx
    //清空画布
    ctx.clearRect(0,0,320,320)
    //使用双重for循环绘制8x8的地图
    for(var i=0;i<8;i++){
      for(var j=0;j<8;j++){
        //默认是道路
        let img ='ice'
        if(map[i][j]== 1){
          img='stone'
        } else 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( )
  },
```

17、在 game.js 的 onload 两数中创建画布上下文,并依次调用自定义函数 initMap 和drawCanvas。对应的 JS (pages/game/game.js)代码片段添加如下:

```
onLoad:function(options){
    //获取关卡
    let level= options.level
    //更新页面关卡标题
```

```
    this.setData({
      level:parseInt(level)+ 1
    })
    //创建画布上下文
    this.ctx = wx.createCanvasContext('myCanvas')
    //初始化地图数据
    this.initMap(level)
    //绘制画布内容
    this.drawCanvas( )
  },
```

当前效果如图：



18、方向键逻辑实现修改 game.wxml页面中的4个方向键 `<button>` ，为其绑定点击事件。WXML (pages/game/game.wxml)代码修改后如下：

```
<!--游戏画布-->
......
<!--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>
<!--“重新开始”按钮-->
......
```

19、在 game.js 文件中添加自定义函数 up、down,left和right,分别用于实现游戏主角(小鸟)在上、下、左、右4个方向的移动，每次点击在条件允许的情况下移动一格。对应的 JS (pages/game/game.js)代码片段添加如下:

```
  //自定义函数--方向键：上
  up:function(){
    //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){
        //更新当前小鸟的坐标
        row = row - 1
      }
      //如果上方是箱子
      else if(box[row-1][col]== 4){
        //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
            //更新当前小鸟的坐标
            row = row - 1
          }
        }
      }
      //重新绘制地图
      this. drawCanvas( )
    }
  },
  //自定义函数--方向键：下
  down:function(){
    //不在最底端才考虑上移
    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){
        //更新当前小鸟的坐标
        row = row + 1
      }
      //如果上方是箱子
      else if(box[row+1][col]== 4){
        //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
            //更新当前小鸟的坐标
            row = row + 1
          }
        }
      }
      //重新绘制地图
      this. drawCanvas( )
    }
  },
  //自定义函数--方向键：左
  left:function(){
    //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){
        //更新当前小鸟的坐标
        col=col-1
      }
      //如果左侧是箱子
      else if(box[row][col-1]== 4){
        //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
            //更新当前小鸟的坐标
            col=col-1
          }
        }
      }
      //重新绘制地图
      this.drawCanvas()
    }
  },
  // 自定义函数--方向键：右
  right:function(){
    // 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){
        // 更新当前小鸟的坐标
        col = col + 1;
      }
      // 如果右侧是箱子
      else if(box[row][col+1] == 4){
        // 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;  // 箱子右移一格
            box[row][col+1] = 0;  // 原箱子位置清空
            // 更新当前小鸟的坐标
```

```
            col = col + 1;
          }
        }
      }
      // 重新绘制地图
      this.drawCanvas();
    }
  },
```

20、在game.js文件中添加自定义函数 isWin，判断逻辑是只要有一个箱子没有在终点位置就判断游戏尚未成功，用于判断游戏是否已经成功对应的 JS (pages/game/game.js)代码片段添加如下:

```
//自定义函数--判断游戏是否成功
  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 false,表示游戏尚未成功
          return false
        }
      }
    }
    //返回 true,表示游戏成功
    return true
  },
```

21、在game.js中添加自定义函数checkWin,要求一旦游戏成功就弹出提示对话框对应的 JS (pages/game/game.js)代码片段修改如下:

```
//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
      })
    }
  },
```

22、最后在game.js的4个方向键函数中追加关于游戏成功判断的雨数,这里以down函数为例,对应的 JS (pages/game/game.js)代码片段修改如下:

```
//自定义函数--方向键：下
  down:function(){
    //不在最底端才考虑上移
    if(row<7){
    //如果下方不是墙或箱子,可以移动小鸟
      ......
      //如果上方是箱子
      ......
      //重新绘制地图
      this. drawCanvas( )
```

```
      //检查游戏是否成功
      this.checkWin()
    }
  },
```

游戏成功画面如图：

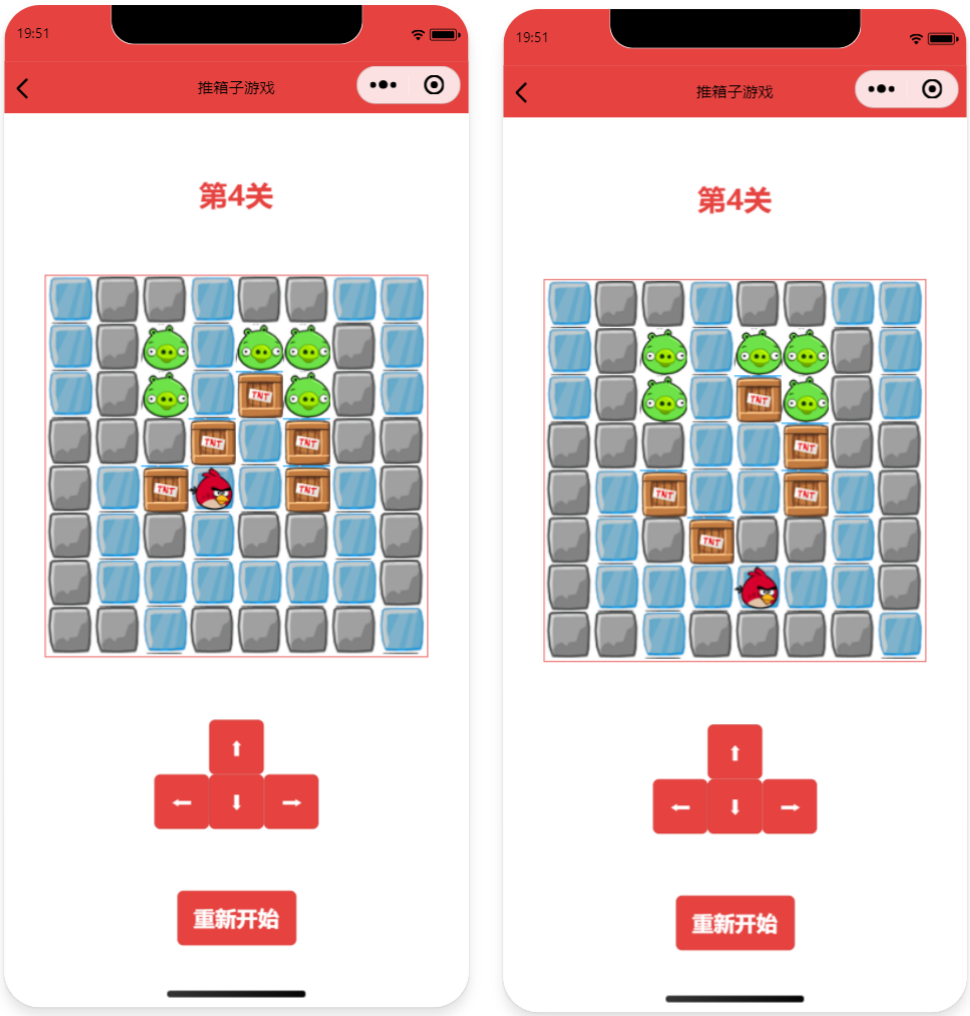
23、修改 game.wxml代码，为“重新开始”按钮追加自定义函数的点击事件，WXML (pages/game/game.wxml)代码片段修改如下：

```
<!--pages/game/game.wxml-->
<view class='container'>
  ......
  <!--“重新开始”按钮-->
  <button type='warn' bindtap='restartGame'>重新开始</button>
</view>
```

24、在game.js文件中添加restartGame函数,用于重新开始游戏。对应的 JS (pages/game/game.js)代码片段添加如下:

```
//自定义函数--重新开始游戏
  restartGame: function(){
    //初始化地图数据
    this.initMap(this.data.level-1)
    //绘制画布内容
    this.drawCanvas( )
  },
```

效果如图：



三、程序运行结果


19:51
推箱子游戏
游戏选关
第1关
第2关
第3关
第4关


19:51
推箱子游戏
第4关
重新开始

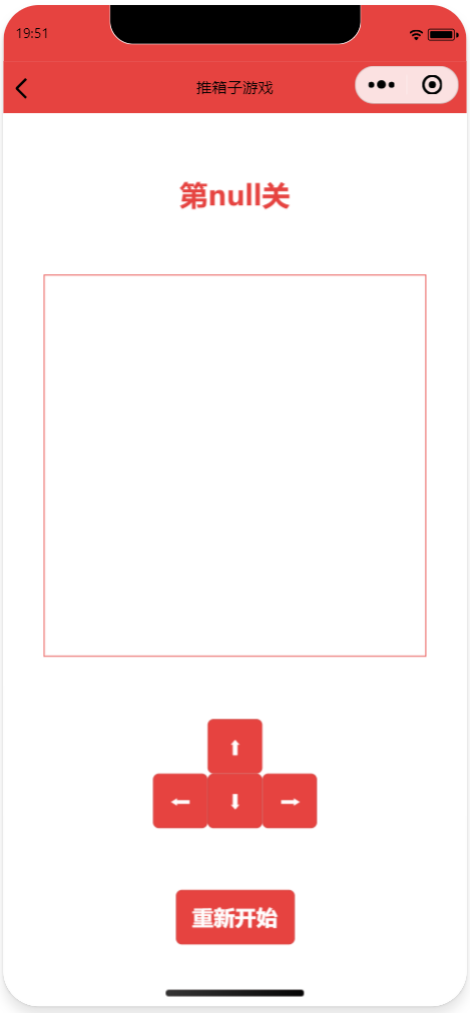

19:51
推箱子游戏
第4关
重新开始


19:51
推箱子游戏
第1关
恭喜
游戏成功!
确定
重新开始

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1：选择关卡后显示错误，地图也无法加载出来。如下：



解决：发现是跳转链接格式错误

这是原来错误的，链接中多了一个空格：

```
// 自定义函数--游戏选关
 chooseLevel:function(e){
   let level = e.currentTarget.dataset.level
   wx.navigateTo({
     url:'../game/game?level= ' + level
   })
 },
```

正确代码如下：

```
// 自定义函数--游戏选关
 chooseLevel:function(e){
   let level = e.currentTarget.dataset.level
   wx.navigateTo({
     url:'../game/game?level=' + level//删掉多余空格
   })
 },
```

收获和体会：

通过本次推箱子游戏开发实验，我不仅系统掌握了微信小程序从页面搭建、数据传递到画布绘制的完整开发流程，更在解决实际问题中深化了对小程序生态的理解。从最初因 URL 参数多空格导致关卡跳转异常，到处理画布绘制中数组越界问题，每一次调试让我对实验有了更深的理解。同时，将公共数据抽离到 `utils/data.js` 实现模块化管理、在 `onLoad` 中初始化画布上下文并调用地图渲染，也让我理解了小程序生命周期与模块化开发的重要性。