2023年夏季《移动软件开发》实验报告

姓名：檀宗晗 学号：21020007087

姓名和学号?	**檀宗晗，21020007087**
本实验属于哪门课程?	中国海洋大学23夏《移动软件开发》
实验名称?	实验4：拼图游戏
博客地址?	https://www.cnblogs.com/-tcxm
Github仓库地址?	https://github.com/tzhcyd/class.git （dev分支）

（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、综合应用所学知识创建完整的拼图游戏项目；2、熟练掌握

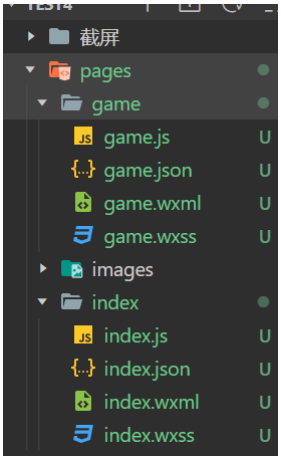
二、实验步骤

1、页面配置

按要求删除文件，并补全page与app函数

添加images文件夹并存储图片

新增game目录来进行游戏页面的设计



2、视图设计

2.1导航栏设计

在app.json添加代码如下：

```
    "pages":[
    "pages/index/index",
    "pages/game/game"
  ],
  "window":{
    "navigationBarBackgroundColor": "#E64340",
    "navigationBarTitleText": "拼图游戏",
    "navigationBarTextStyle":"white"
  },
```

可将所有页面导航栏标题文本为“拼图游戏”，背景颜色为珊瑚红，字体颜色为白色，效果如图：

2.2页面设计

2.2.1公共样式设计

app.wxss代码如下：

```
/*页面容器样式*/
.container {
    height : 100vh;
    color : #E64340;
    font-weight: bold;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: space-evenly;
```

```
}

/*顶端标题样式*/
.title {
    font-size: 18pt;
}
```

2.2.2首页设计

包含两部分内容：标题和关卡列表

wxml代码如下：

```
<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 = '/pages/images/pic01.jpg'></image>
            <text>第1关</text>
        </view>
    </view>
</view>
```

wxss代码如下：

```
/*关卡列表区域*/
.levelBox{
    width: 100%;
}
/*单个关卡区域*/
.box{
    width : 50%;
    float : left;
    margin : 25rpx 0;
    display: flex;
    flex-direction: column;
    align-items: center;
}

/*选关图片*/
image{
    width : 260rpx;
    height: 260rpx;
}
```

可得效果图：（由于尚未获得关卡数据，暂时无法显示完整的关卡列表，只显示标题与临时关卡）

2.2.3游戏页面设计

wxml代码如下：

```
<view class = 'container'>
    <!--提示图区域-->
    <view class = 'title'>提示图</view>
    <image src = '/pages/images/pic01.jpg'></image>

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

    <!--“重新开始”按钮-->
    <button type = 'warn' >重新开始</button>
</view>
```

wxss代码如下：

```
/*提示图样式*/
image{
    width: 250rpx;
    height: 250rpx;
}

/*画布样式*/
canvas{
    border: 1rpx solid;
    width: 300px;
    height : 300px;
}
```

效果如图：（由于尚未获得游戏数据，故无法根据用户点击的关卡入口显示对应的游戏内容）

3、逻辑实现

3.1首页逻辑

3.1.1关卡列表展示

wx:for可以控制属性绑定一个数组，可以使用数组中各项的数据重复渲染该数组，默认数组的当前项的下表变量名默认为index，数组当前项的变量名默认为item；

wx:key可以指定列表中项目的唯一标识符

wxml代码如下：

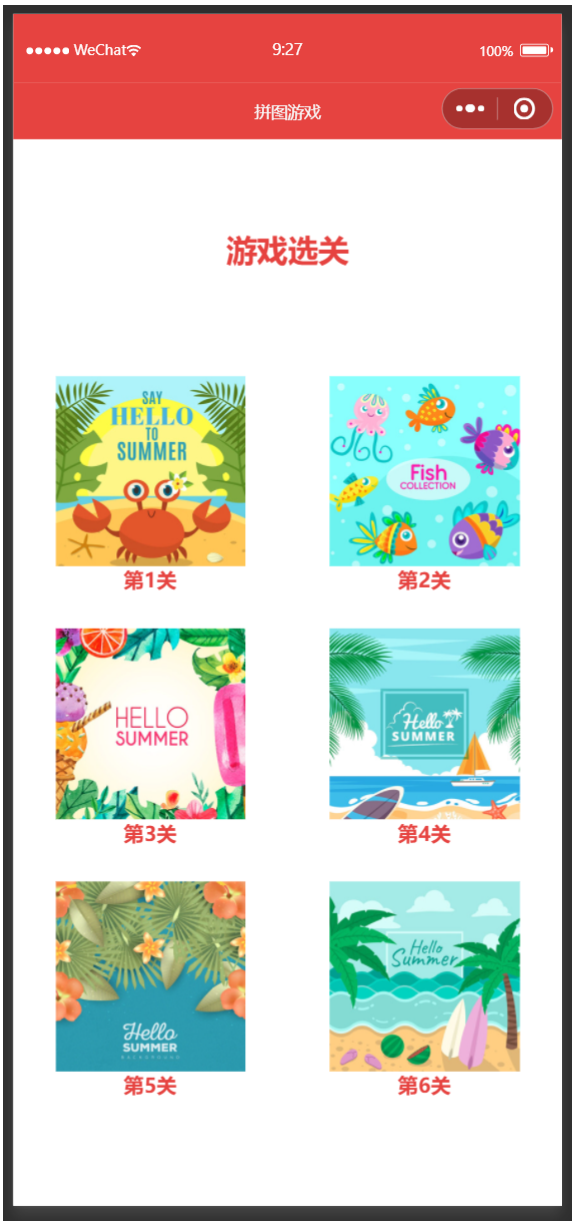
```
<view class = 'container'>
    <!--标题-->
    <view class = 'title'>游戏选关</view>

    <!--关卡列表-->
    <view class = 'levelBox'>
        <view class = 'box' wx:for = '{{levels}}' wx:key="levels{{index}}">
            <image src = '/pages/images/{{item}}'></image>
            <text>第{{index + 1}}关</text>
        </view>
    </view>
</view>
```

js文件中数据追加levels数组如下：

```
 levels:[
        'pic01.jpg',
        'pic02.jpg',
        'pic03.jpg',
        'pic04.jpg',
        'pic05.jpg',
        'pic06.jpg',
    ]
```

效果如图：



3.1.2点击跳转游戏页面

通过bindtap事件可以使用户点击图片时实现跳转，并使用data-level属性携带了关卡图片信息。

wxml代码如下：

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

js新增函数如下，wx.navigateTo即可实现页面的跳转，并携带了关卡图片数据。

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

但此时仍需在游戏页面接受关卡信息才可以正确显示游戏画面。

3.2游戏页逻辑

3.2.1显示提示图

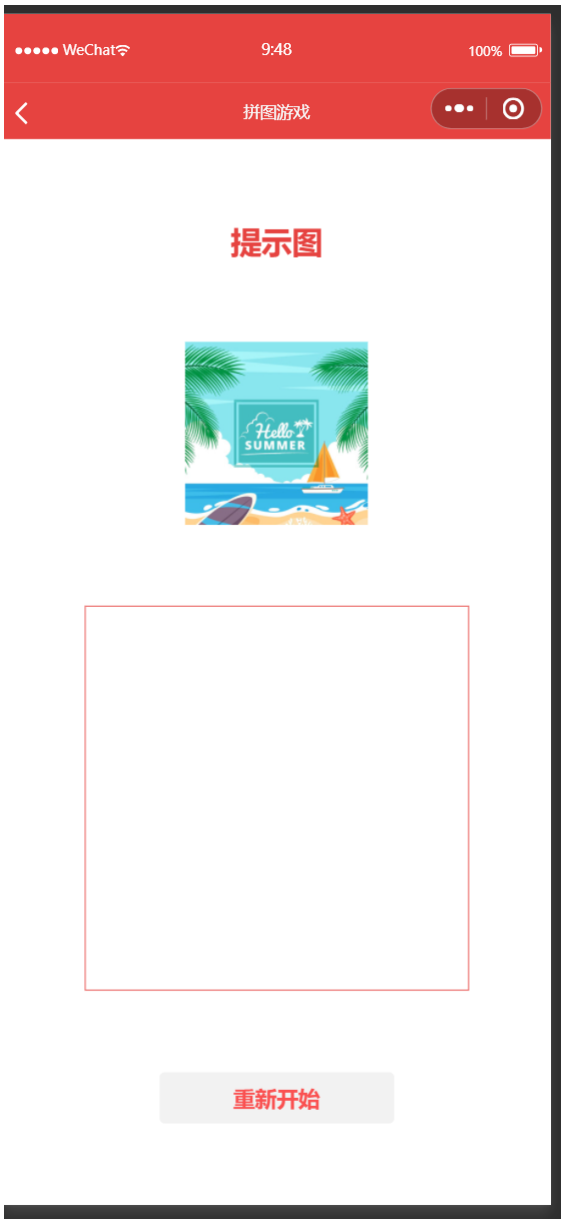
wxml代码如下：

```
<!--提示图区域-->
    <view class = 'title'>提示图</view>
    <image src = '{{url}}'></image>
```

onLoad函数修改如下，可以通过options参数将index页面中的图片url参数传到当前页面，即可完成修改图片地址，并通过动态数据将图片显示出来。

```
 onLoad: function(options) {
        //更新图片路径地址
         let url = '/pages/images/' + options.level
        //更新提示图的地址
        this.setData({url : url})
```

效果如图：



3.2.2游戏逻辑实现

1）准备工作

在game.js文件中记录游戏初始数据，用012来代表方块的初始位置，代码如下：

```
//方块的初始位置
var num =[
    ['00', '01', '02'],
    ['10', '11', '12'],
    ['20', '21', '22']
]
//方块的宽度
var w = 100
//图片的初始地址
var url ='/pages/images/pic01.jpg'
```

2）初始化游戏拼图

考虑从空白方块所在的位置入手，每次随机让他和她周围的临近方块交换位置，这样就可以实现一定可以通过反向移动回到最初的位置。

随机交换100次顺序即可实现打乱方块顺序的要求。

在game.js添加函数shuffle，用于重新开始游戏，代码如下：

```
shuffle : function(){
        //先令所有方块回归初始位置
        num = [
            ['00', '01', '02'],
            ['10', '11', '12'],
            ['20', '21', '22']
        ]
        //记录当前空白方块的行和列
        var row = 2
        var col = 2
        //打乱方块顺序100次
        for(var i = 0; i < 100; i++){
            //随机产生其中一个方向：上（0）、下（1）、左（2）、右（3）
            var direction = Math.round(Math.random()*3)
            //上：0
            if(direction == 0){
                //空白方块不在最上面一行
                if(row != 0){
                    //交换位置
                    num[row][col] = num[row - 1][col]
                    num[row - 1][col] = '22'
                    //更新空白方块的行
                    row -= 1
                }
            }
            //下：1
            else if(direction == 1){
                //空白方块不在最下面一行
                if(row != 2){
                    //交换位置
                    num[row][col] = num[row + 1][col]
                    num[row + 1][col] = '22'
                    //更新空白方块的行
                    row += 1
                }
            }
            //左：2
            else if(direction == 2){
                //空白方块不在最左侧
                if(col != 0){
                    //交换位置
                    num[row][col] = num[row][col-1]
                    num[row][col-1] = '22'
                    //更新空白方块的行
                    col -= 1
```

```
                }
            }
             //右：3
             else if(direction == 3){
                //空白方块不在最左侧
                if(col != 2){
                    //交换位置
                    num[row][col] = num[row][col+1]
                    num[row][col+1] = '22'
                    //更新空白方块的行
                    col += 1
                }
            }
        }
    },
```

上述代码使用for循环进行了100次打乱，使用Math.random()方法从上下左右四个方向随机产生一个，若符合可以交换的条件则进行交换。Math.round()函数可以讲一个数字四舍五入到最接近的整数，Math.random()函数可生成0.0-1.0的数，乘3取整即可实现随机产生四个数的操作，代表四个方向。

在game.js中添加函数drawCanvas，用于将打乱后的图片方块绘制到画布上，ctx.clearRect()可以把像素设置为透明以达到擦除一个矩形区域的目的。ctx.drawImage可以将图片裁剪，并位于画布的j * w,i * w位置，并确定所使用图像的宽度与高度w，对应的js代码如下：

```
/**
     * 自定义函数 -- 绘制画布内容
     */
    drawCanvas : function(){
        let ctx = this.ctx
        //清空画布
        ctx.clearRect(0,0,300,300)
        //使用双重for循环绘制3*3的拼图
        for(var i = 0; i < 3; i++){
            for(var j = 0; j < 3; j++){
                if(num[i][j] != '22'){
                    //获取行和列
                    var row = parseInt(num[i][j] / 10)
                    var col = num[i][j] % 10
                    //绘制方块
                    ctx.drawImage(url, col*w, row*w,w,w,j*w,i*w,w,w)
                }
            }
        }
        ctx.draw()
    },
```

在game.js的onLoad函数中调用自定义函数shuffle与drawCanvas，在做实验的时候发现wx.createCanvasContext函数已经被弃用，但是现在仍然可以继续使用，就没有对此进行修改，但自己了解了一下可以使用Canvas进行替换，对应的js代码如下：

```
onLoad: function(options) {
      //更新图片路径地址
       let url = '/pages/images/' + options.level
      //更新提示图的地址
      this.setData({url : url})
      //创建画布上下文
      this.ctx = wx.createCanvasContext('myCanvas')
      //打乱方块顺序
      this.shuffle()
      this.drawCanvas()
 },
```

可得效果如图：

3）移动被点击的方块

修改画布组件（canvas），为其绑定触摸事件，修改代码如下：

```
<!--游戏画布-->
 <canvas canvas-id="myCanvas" bindtouchstart="touchBox"></canvas>
```

在game.js文件中添加自定义函数touchBox，用于实现图片方块的移动，代码如下：

```
/**
 * 自定义函数 -- 监听点击方块事件
 */
    touchBox : function(e){
        //如果游戏已经成功，不做任何操作
        if(this.data.isWin){
            return
        }
         //获取被点击方块的坐标x和y
          var x = e.changedTouches[0].x
          var y = e.changedTouches[0].y
          var row = parseInt(y / w)
          var col = parseInt(x / w)
          //如果点击的不是空白位置
          if(num[row][col] != '22'){
              //尝试移动方块
              this.moveBox(row,col)
              //重新绘制画布内容
              this.drawCanvas()
              //判断游戏是否成功
              if(this.isWin()){
                  //在画面上绘制提示语句
                  let ctx = this.ctx
                  //绘制完整图片
                  ctx.drawImage(url, 0, 0)
                  //绘制文字
                  ctx.setFillStyle('#e64340')
                  ctx.setTextAlign('center')
                  ctx.setFontSize(60)
                  ctx.fillText('游戏成功',150,150)
                  ctx.draw()
              }
          }
    },
/**
 * 自定义函数 -- 移动被点击的方块
 */
    moveBox : function(i, j){
        //情况1：如果被点击的方块不在最上方，检查可否上移
        if(i>0){
            //如果方块上是空
            if(num[i-1][j]=='22'){
                //交换位置
                num[i-1][j] = num[i][j]
                num[i][j] = '22'
                return
```

```
            }
        }
        //情况2：如果被点击的方块不在最下方，检查可否下移
        if(i<2){
            //如果方块下是空
            if(num[i+1][j]=='22'){
                //交换位置
                num[i+1][j] = num[i][j]
                num[i][j] = '22'
                return
            }
        }
        //情况3：如果被点击的方块不在最左方，检查可否左移
        if(j>0){
            //如果方块左是空
            if(num[i][j-1]=='22'){
                //交换位置
                num[i][j-1] = num[i][j]
                num[i][j] = '22'
                return
            }
        }
           //情况4：如果被点击的方块不在最右方，检查可否右移
           if(j<2){
            //如果方块左是空
            if(num[i][j+1]=='22'){
                //交换位置
                num[i][j+1] = num[i][j]
                num[i][j] = '22'
                return
            }
        }
    },
```

现在即可实现点击图片即可进行移动。

3.2.3判断游戏成功

在game.js文件中data中添加初始数据isWin，用来标记游戏成功与否，若为false即为还未成功，当成功时会重置为true，并且在js文件中添加自定义函数isWin来判断游戏是否成功，判断的原理也比较简单，只需要将当前方块的i乘10与j相加，再与标准大小进行比对即可，对应的代码如下：

```
/**
 * 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 < 3; i++){
            for(var j = 0; j < 3; j++){
                //如果有方块位置不对
                if(num[i][j] != i*10+j){
                    return false
```

```
                }
            }
        }
        //更新成功状态
        this.setData({isWin : true})
        return true
    },
```

然后修改game.js中的touchBox函数，每一次被移动都要追加对于游戏是否成功的判断，增添的代码如下：

```
//如果游戏已经成功，不做任何操作
        if(this.data.isWin){
            return
        }
        //判断游戏是否成功
              if(this.isWin()){
                  //在画面上绘制提示语句
                  let ctx = this.ctx
                  //绘制完整图片
                  ctx.drawImage(url, 0, 0)
                  //绘制文字
                  ctx.setFillStyle('#e64340')
                  ctx.setTextAlign('center')
                  ctx.setFontSize(60)
                  ctx.fillText('游戏成功',150,150)
                  ctx.draw()
              }
```

游戏成功的效果图如下：

3.2.4游戏重新开始

修改game.wxml代码，为"重新开始"的按键追加自定义函数的点击事件，代码如下：

```
<!--“重新开始”按钮-->
    <button type = 'warn' bindtap = 'restartGame'>重新开始</button>
```

在game.js中添加restartGame函数，用于重新开始游戏，只需要重新打乱方块顺序并且重新绘制画布内容即可实现重新开始游戏的目标，代码如下：

```
/**
 * 自定义函数--重新开始游戏
 */
    restartGame : function(){
        this.setData({isWin:false})
        //打乱方块顺序
        this.shuffle()
        //绘制画布内容
        this.drawCanvas()
    },
```

现在即可实现该游戏的全部功能。

三、程序运行结果

游戏主页面：

为了方便测试，将图片移动次数改为1，在进行游戏前图片如下：

移动后（即完成小游戏）：

四、问题总结与体会

本次实验让我学习到了canvas组件的用法，canvas组件需要一个唯一标识符canvas-id，通过自己的学习也知道了如今最新的canvas组件可以通过SelectorQuery来选择上一步的canvas，这样就可以获取到canvas对象。但通过我的学习我也知道了canvas组件还有很多方法，需要在课下的时候自己进行学习。

在实验过程中也出现了一些小问题，经常会发生由于大小写拼写错误以及多一个空格导致编译失败的事件，要注意这些小问题的出现。