2024年夏季《移动软件开发》实验报告

姓名：缪纬韬 学号：22020007160

姓名和学号	缪纬韬，22020007160
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	http://t.csdnimg.cn/3VnwH
Github仓库地址	https://github.com/spchara/remote-software-develop-lab.git

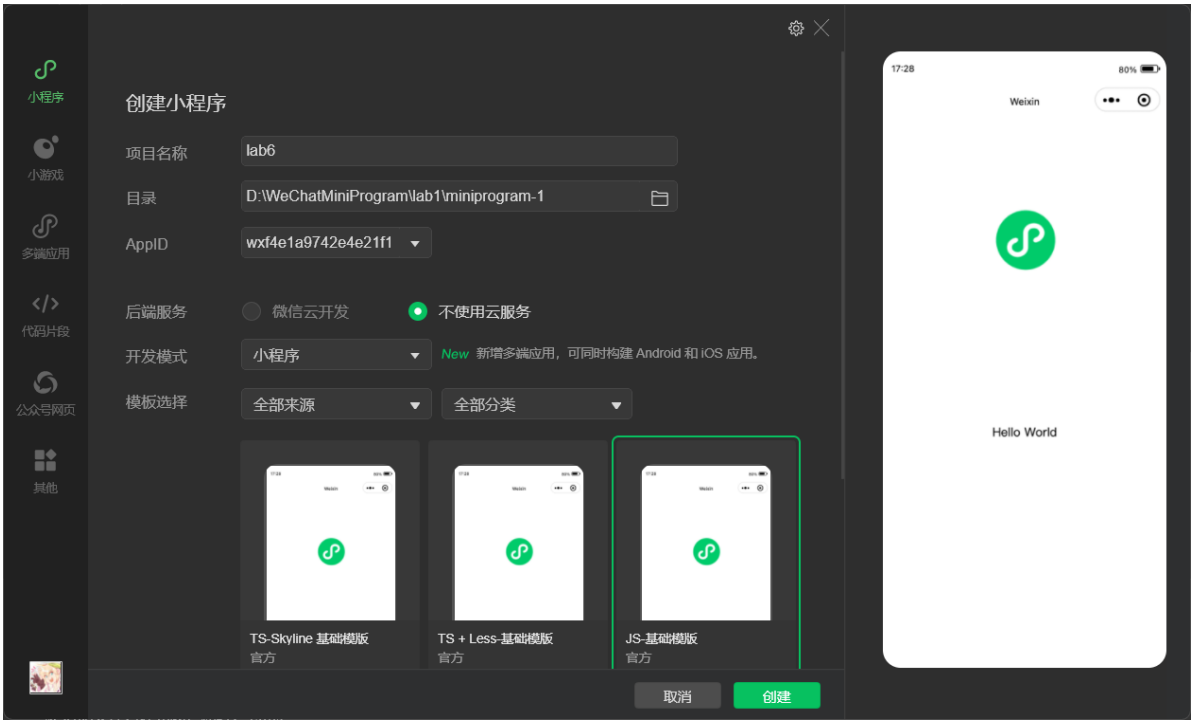
一、实验目标

1. 综合应用所学的知识创建完整的推箱子游戏
2. 熟练掌握<canvas>和绘图API

二、实验步骤

1 创建项目并完成前期准备

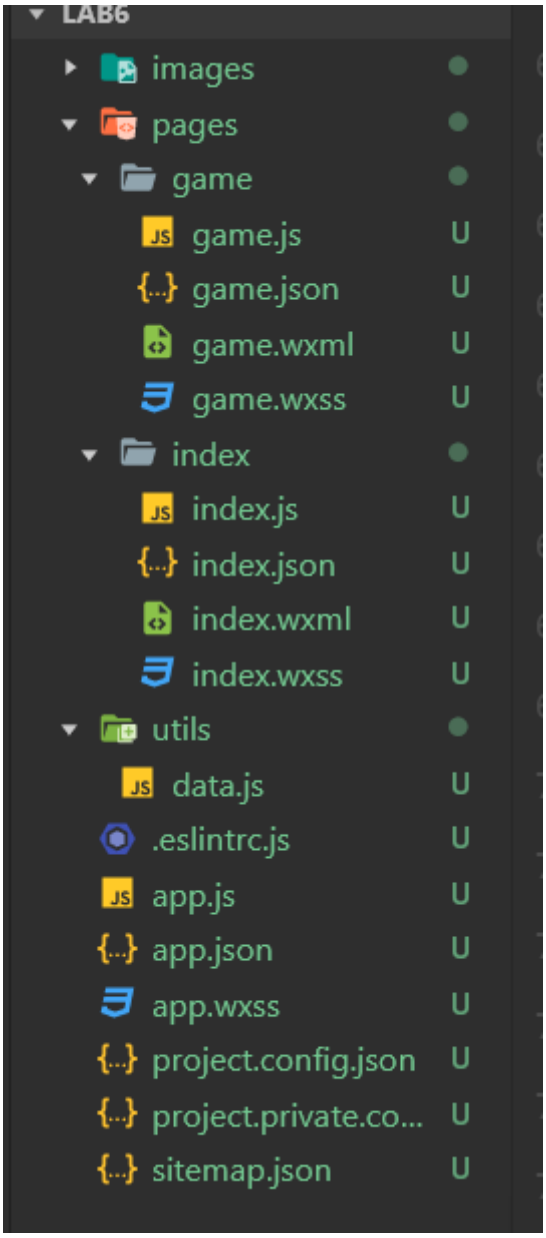
创建空白项目



在根目录下创建`images`文件夹和`utils`文件夹，将老师提供的图片放置在`images`下，并在`utils`目录下创建文件`data.json`

修改`app.json`中的页面路径，由于我们需要第二个游戏页面，所以添加路径`pages/game/game`

```
"pages": [
  "pages/index/index",
  "pages/game/game"
],
```



2 导航栏设计

一如既往，修改`app.json`

```
"window": {
  "navigationBarTextStyle": "white",
  "navigationBarTitleText": "推箱子游戏",
  "navigationBarBackgroundColor": "#E64340"
},
```

3 页面设计

3.1 公共样式设计

在`app.wxss`中，完成程序通用的页面容器、标题样式设计

代码如下：

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

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

3.2 首页设计

首页包含的功能有标题和关卡列表

代码如下，css在文后，提前写出循环，预留出关卡列表的位置：

```
<!--index.wxml-->
<scroll-view class="scrollarea" scroll-y type="list">
  <view class="container">
    <view class="title">游戏选关</view>
    <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>
</scroll-view>
```

预期效果如下：

游戏选关

第1关

第2关

第3关

第4关

3.3 游戏界面设计

游戏界面需要标题、游戏画面、方向键、和重新开始的按钮，依次排开即可

```
<view class="container">
  <view class="title">第{{level}}关</view>

  <canvas canvas-id="mycanvas"></canvas>

  <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 type="warn" bindtap="restartGame">重新开始</button>
</view>
```

4 业务逻辑实现

4.1 公共逻辑

公共逻辑部分很简单，只需要把每个关卡的地图数据以矩阵的形式保存在里面，并提供返回这些数据的接口即可

```
//1为墙，2为路，3为终点，4为箱，5为人物，0为地图边缘

var map1 = [
    ...略...
];

var map2 = [
    ...略...
];

// 关卡 3
var map3 = [
    ...略...
];

// 关卡 4
var map4 = [
    ...略...
];

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

```

4.2 首页逻辑

为了显示关卡缩略图，首先在data里录入信息

```
data: {
   levels: [
     'level01.png',
     'level02.png',
     'level03.png',
     'level04.png'
   ]
 },
```

然后给图片添加点击跳转即可，并携带关卡号信息即可

```
 chooseLevel:function(e){
   let level = e.currentTarget.dataset.level;
   wx.navigateTo({
     url: "../game/game?level="+level
   })
 },
```

4.3 游戏逻辑

游戏逻辑较为复杂，包含四个部分

- 画面绘制

  在页面加载时，需要加载地图数据，我们用两个矩阵来保存，一个map，一个box，方便判断移动，以及记录小鸟的当前位置，再根据所保存的地图数据，用`images/icon`中的图标来绘制画面。

```
var data = require('../../utils/data.js')
var map = [
全0
]

var box=[
全0
]

var w = 40
var row = 0
var col = 0

Page({
    initMap: function(level){
    let mapData = data.maps[level];
    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;
          row = i;
          col = j;
        }
      }
    }
    },

    drawCanvas:function(){
    let ctx = this.ctx;
    ctx.clearRect(0,0,320,320);
    for(var i=0;i<8;i++){
      for(var j=0;j<8;j++){
        let img = 'ice'
        if(map[i][j]==1){
          img='stone'
        }
        else if(map[i][j] == 3){
          img ='pig'
        }
        ctx.drawImage("/images/icons/"+img+".png", j*w,i*w,w,w);

        if(box[i][j]==4){
          ctx.drawImage("/images/icons/box.png", j*w,i*w,w,w);
        }
      }
    }
    ctx.drawImage("/images/icons/bird.png", col*w,row*w,w,w);
    console.log("row:"+row+"col:"+col);
    ctx.draw();
  },


  onLoad(options) {
    let level=options.level;
    this.setData({
      level:parseInt(level)+1
    })
    this.ctx = wx.createCanvasContext('mycanvas');
    this.initMap(level);
    this.drawCanvas();
  },
})
```

- 移动逻辑

  移动包含四个方向，基本逻辑就是当被调用时，依次判断：

  - 是否在地图边缘？
    - 上方是否是墙或箱子？
      - 若是，则判断是否是墙
        - 是墙则不能移动
      - 是箱子则判断箱子是否可以被移动，过程和鸟的判断基本一致

        - 若可以，则移动箱子和鸟
    - 否则直接移动

写出四个方向的逻辑即可，这里只展示向上的

```
up:function(){
   //不在最上层
   if(row>0){
     //如果上面不是墙或箱子

     if(map[row-1][col] != 1 && box[row - 1][col] !=4){
       //移动
       row = row -1
     }
     //如果上面是箱子
     else if(box[row - 1][col] ==4){
       //箱子可以推嘛？
       if(row-1>0){
         //如果可以
         if(map[row-2][col]!=1 && box[row-2][col]!=4){
           box[row-2][col]=4
           box[row-1][col]=0
           row = row-1
         }
       }
     }
   }
   this.drawCanvas();
   this.checkWin();
 },
```

- 判断游戏成功

  通过box矩阵中的4和map矩阵中的3是否全部重合，若有一个不重合，即说明没有成功

  然后在四向移动的逻辑中调用这个检查即可

```
 isWin:function(){
   for(var i=0;i<8;i++){
     for(var j=0;j<8;j++){
       if(box[i][j] == 4 && map[i][j]!=3){
         return false;
       }
     }
   }
   return true;
 },

 checkWin:function(){
   if(this.isWin()){
     wx.showModal({
       title:"恭喜",
       content:"游戏成功！",
       showCancel:false,
     })
   }
 },
```

- 重新开始

  绑定按钮，用最开始的数据，重新调用初始化程序即可

```
restartGame:function(){
  this.initMap(this.data.level -1);
  this.drawCanvas();
},
```

最后将这些功能都绑定到按钮上，逻辑部分就全部完成了

三、程序运行结果

首页

游戏选关

第1关

第2关

第3关

第4关

关卡界面

11:45 100%

推箱子游戏

第1关

↑

← ↓ →

重新开始

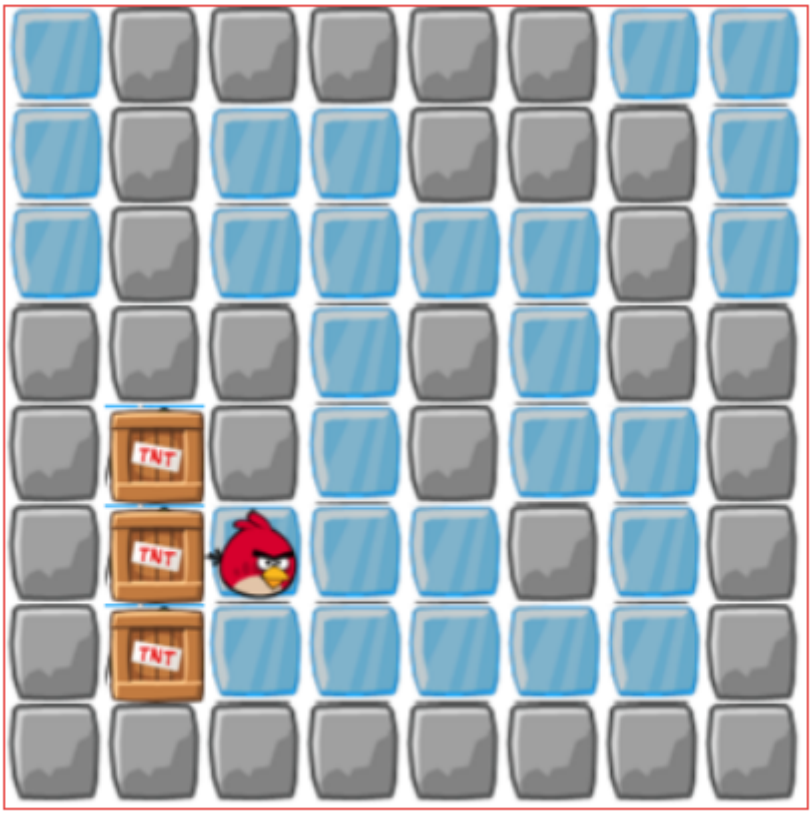
游戏获胜


11:46
100%
推箱子游戏
第1关
恭喜
游戏成功！
确定
↑
←
↓
→
重新开始

11:46 100%

推箱子游戏

第1关



↑

← ↓ →

重新开始

四、问题总结与体会

问题：

- Q：createCanvasContext方法被提示版本过低，建议使用其他的，不过还是能用

```
[pages/game/game] [Component] <canvas>: canvas 2d 接口支持同层渲染且性能更佳，建议切换使用。详见文档 https://developers.weixin.qq.com/miniprogram/dev/component/canvas.html#Canvas-2D-%E7%A4%BA%E4%BE%8B%E4%BB%A3%E7%A0%81
```

体会

这次实验中我第一次实现了微信小程序里canvas组件接口的学习和使用，完成一个小游戏还是挺有意思的