**学年论文**

**（课程论文、课程设计）**

**（2019-2020学年第一学期）**

**题　　目： 太空版植物大战僵尸前端页游**

**2019年 1月 10日**

目录

[1 绪论 1](#_Toc29459993)

[1.1 开发背景 1](#_Toc29459994)

[1.2 系统开发平台 1](#_Toc29459995)

[1.3 系统开发环境 1](#_Toc29459996)

[1.3.1 硬件环境 2](#_Toc29459997)

[1.3.2 软件环境 2](#_Toc29459998)

[2 系统分析 3](#_Toc29459999)

[2.1系统的需求分析 3](#_Toc29460000)

[2.2系统可行性分析 3](#_Toc29460001)

[3系统概要设计 5](#_Toc29460002)

[3.1 设计思想 5](#_Toc29460003)

[3.2设计原则 5](#_Toc29460004)

[3.3业务流程分析 5](#_Toc29460005)

[4系统详细设计 7](#_Toc29460006)

[4.1 系统功能结构图 7](#_Toc29460007)

[4.2 系统功能设计 7](#_Toc29460008)

[4.3 系统数据库设计 8](#_Toc29460009)

[4.3.1 E-R模型 8](#_Toc29460010)

[4.3.2 数据库表实现 9](#_Toc29460011)

[5系统开发与实现 10](#_Toc29460012)

[5.1 登录界面 10](#_Toc29460013)

[5.1.1验证码随机生成功能代码 10](#_Toc29460014)

[5.1.2用户名及密码验证功能： 12](#_Toc29460015)

[5.2 游戏功能具体实现代码 12](#_Toc29460016)

[5.2.1实现物体的拖拽动态特效及创建道具函数： 12](#_Toc29460017)

[5.2.2实现外星人的动态下滑特效： 13](#_Toc29460018)

[5.2.3气泡的螺旋升腾特效 14](#_Toc29460019)

[5.2.4实现外星人攻击建筑物时的冒烟效果 15](#_Toc29460020)

[5.2.5实现外星人的攻击冲击波 16](#_Toc29460021)

[6系统测试 17](#_Toc29460022)

[7 总结与展望 22](#_Toc29460023)

[7.1总结 22](#_Toc29460024)

[7.2展望 22](#_Toc29460025)

[参考文献 23](#_Toc29460026)

# 1 绪论

## 开发背景

这是一个经济社会快速发展的年代,人们的生活节奏越来越快,生活中的各种压力也越来

越大。而电子游戏的出现,则为身处重重压力下的现代人提供了一种很好的娱乐休闲方式。

近年来,我国的游戏产业取得了长足的进步,获得了比较快的发展。中国游戏产业总体上朝着一个更加健康、更加快速和有序的方向发展，产业规模、市场规模进一步扩大，游戏产业的自主创新能力不断增强，行业活力盎然。《2009年中国游戏产业报告》相关数据显示：2009年中国网络游戏实际销售收入为256.2亿元，比2008年增长了39.4% 。

小游戏作为电子游戏家族中的一个独特的成员，在其中扮演着重要的角色。 小游戏是相对于那些体积庞大、动辄需要几百兆甚至几个G安装空间，需要较高的CPU、内存、显卡性能支持的大型单机游戏或者大型网络游戏而言的，泛指所有体积较小，玩法比较简单，对计算机性能要求比较低的游戏，通常这类游戏以有体积较小，玩法比较简单，对计算机性能要求比较低的游戏，通常这类游戏以休闲益智类为主，主要分为单机版和网页版。近年来，小游戏产业发展迅速，各种新颖的小游戏层出不穷，形式五花八门，广受各个年龄段的用户的欢迎，玩家 群迅速增加。本次开发的太空版植物大战僵尸是属于休闲益智型小游戏，提高自身的反应能力，正是迎合了当前游戏产业特别是小游戏产业蓬勃发展的机遇，具备很好的发展前景。

## 系统开发平台

应用采用VUE和WebPack模块打包器开发一款类似于植物大战僵尸的前端游戏

## 1.3 系统开发环境

本游戏采用VUE语言和WebPack模块打包器制作游戏并用bootstarp框架，制作前端模板，采取前后端分离的开发方式，用vue前端+java后端来实现游戏的运行。

### 1.3.1 硬件环境

Window电脑

内存>4G

硬盘容量可用空间大于1GB

网卡显卡驱动正常

### 1.3.2 软件环境

（1）VUE语言和WebPack包管理器

（2）eclipse Java EE 2019版本

（3）tomcat8.0及以上版本

# 2 系统分析

## 2.1系统的需求分析

功能需求:

1)界面简洁。

2)游戏规则合理。

在本游戏中，用户需要登录并输入验证码后才能进入游戏界面进行游戏操作，在用户输入账户名，密码和验证码后，程序对其进行验证，验证成功过则进入游戏界面，失败则刷新页面。在本游戏中，一系列外星飞船从天而降入侵地球，为了保护地球，玩家需要使用各种道具防止外星飞船落入底部的地球。这些道具可以是箱子，也可以是卫星。当卫星会产生一系列能量，点击这些能量后，玩家可以获得积分，一旦积分达到要求，玩家就可以选择炮台，炮台可以发射子弹，一旦子弹打中外星飞船，飞船就会从界面上消失，游戏的玩法其实和植物大战僵尸是如出一辙。

需求环境：

本课程设计需要的设备为硬件要求和软件配置要求具体要求如下：

1硬件配置：一台计算机

2软件配置：WINDOWS、MY SQL, exclipse

## 2.2系统可行性分析

这是一个经济社会快速发展的年代,人们的生活节奏越来越快,生活中的各种压力也越来越大。而电子游戏的出现,则为身处重重压力下的现代人提供了一种很好的娱乐休闲方式。

近年来,我国的游戏产业取得了长足的进步,获得了比较快的发展。中国游戏产业总体上朝着一个更加健康、更加快速和有序的方向发展，产业规模、市场规模进一步扩大，游戏产业的自主创新能力不断增强，行业活力盎然。《2009年中国游戏产业报告》相关数据显示：2009年中国网络游戏实际销售收入为256.2亿元，比2008年增长了39.4% 。

小游戏作为电子游戏家族中的一个独特的成员，在其中扮演着重要的角色。 小游戏是相对于那些体积庞大、动辄需要几百兆甚至几个G安装空间，需要较高的CPU、内存、显卡性能支持的大型单机游戏或者大型网络游戏而言的，泛指所有体积较小，玩法比较简单，对计算机性能要求比较低的游戏，通常这类游戏以有体积较小，玩法比较简单，对计算机性能要求比较低的游戏，通常这类游戏以休闲益智类为主，主要分为单机版和网页版。近年来，小游戏产业发展迅速，各种新颖的小游戏层出不穷，形式五花八门，广受各个年龄段的用户的欢迎，玩家 群迅速增加。本次开发的太空版植物大战僵尸是属于休闲益智型小游戏，提高自身的反应能力，正是迎合了当前游戏产业特别是小游戏产业蓬勃发展的机遇，具备很好的发展前景。

# 3系统概要设计

## 3.1 设计思想

本游戏画面精致，操作人性化，非常适合各年龄段的玩家，休闲娱乐用。

在本游戏中，用户需要登录并输入验证码后才能进入游戏界面进行游戏操作，在用户输入账户名，密码和验证码后，程序对其进行验证，验证成功过则进入游戏界面，失败则显示验证失败！字样。在本游戏中，一系列外星飞船从天而降入侵地球，为了保护地球，玩家需要使用各种道具防止外星飞船落入底部的地球。这些道具可以是箱子，也可以是卫星。当外星飞船被成功阻挡时，界面会弹出一系列奖章，点击这些奖章后，玩家可以获得积分，一旦积分达到要求，玩家就可以选择炮台，炮台可以发射子弹，一旦子弹打中外星飞船，飞船就会从界面上消失，游戏的玩法其实和植物大战僵尸是如出一辙。

## 3.2设计原则

游戏的一个可玩性在于，让玩家在一种战略平衡中做抉择。我们在游戏开始时让玩家拥有120单位的能量，玩家可以采取以下几种步骤来开展游戏：

建筑一个卫星30E，一个炮台80E，然后还剩10E。然后等300毫秒后收集第一个能量泡，从而获得足够能源建造其他卫星或炮台。

连续建造两个卫星60E，剩下60E，等300毫秒后获取两个能量泡，从而获得总共260E的能量，然后玩家可以连续建造三个炮台。一下子建造3个卫星90E，还剩30E。等300毫秒后连续收获3个能量泡，使得能源达到330E，然后可以一口气建造4个炮台。一下子建造4个卫星120E,等300毫秒后收获4个能量泡，然后一下子建造5个炮台。

无论玩家作何选择，最终目的是顶住冲击波中的外星人攻击，好的战略必须使得玩家能够迅速的建筑足够多的炮台。如果你一口气建造4个卫星，那么你只能眼睁睁的看着外星人攻打过来，你很可能来不及建造炮台就完蛋了。因此玩家要根据每一轮冲击波中外星人的数量和特性，在资源约束下，选择不同的建造策略，如此一来，我们的游戏就具备即时战略的可玩性。

## 3.3业务流程分析

1. 登录界面

可以为用户提供账户的注册登录功能，并且提供随机生成的验证码进行验证登录。用户在输入正确的账户名和密码后可以进入游戏开始界面，否则网页底部显示验证失败！账户名或密码错误字样。

1. 物体拖拽动态特效

玩家先在底部的按钮中选择点击某个按钮，点击后图中的黄色方框会显示在页面上，且方框为动态，在指定位置会一放一缩。当玩家选定方块后，点击鼠标，那么在相应方块里就会出现对于的道具。道具的种类是根据玩家在底部点击哪个按钮决定的

1. 能量气泡的螺旋升腾特效

当玩家在页面上建造一个卫星道具后，每隔一段时间就会有一个能量泡从卫星处弹出，然后以固定轨迹慢慢的往顶部升腾，当玩家用鼠标点击能量泡后，玩家就可以获得100点的资源，这些资源可用来购买更高级的道具

1. 外星人的动态下滑特效

游戏开始后，外星人会从网页顶部生成并动态下滑，慢慢接近玩家所部署的道具建筑物。

1. 实现外星人攻击建筑物时的冒烟效果

外星人会使用冲击波攻击建筑物，建筑物在被攻击后会生成冒烟效果。

1. 实现子弹发射击打外星人效果

玩家在部署基本建筑物后，具有攻击功能的建筑物/道具会发射子弹击打外星人。

1. 实现外星人的攻击冲击波

外星人攻击式会生成攻击冲击波。

1. 通关排行榜模块功能及评论模块功能

系统将不同用户产生的游戏数据进行排序并生成排行榜，用户同时也可以对本游戏在评论模块处进行评价

# 4系统详细设计

## 4.1 系统功能结构图

用户

：

图4-1 用户功能用例图

系统：



图4-2 系统功能用例图

## 系统功能设计

(1)登录

可以为用户提供账户的注册登录功能，并且提供随机生成的验证码进行验证登录。用户在输入正确的账户名和密码后可以进入游戏开始界面，否则网页底部显示验证失败！账户名或密码错误字样。

(2)物体拖拽动态特效

玩家先在底部的按钮中选择点击某个按钮，点击后图中的黄色方框会显示在页面上，且方框为动态，在指定位置会一放一缩。当玩家选定方块后，点击鼠标，那么在相应方块里就会出现对于的道具。道具的种类是根据玩家在底部点击哪个按钮决定的

(3)能量气泡的螺旋升腾特效

当玩家在页面上建造一个卫星道具后，每隔一段时间就会有一个能量泡从卫星处弹出，然后以固定轨迹慢慢的往顶部升腾，当玩家用鼠标点击能量泡后，玩家就可以获得100点的资源，这些资源可用来购买更高级的道具

(4)外星人的动态下滑特效

游戏开始后，外星人会从网页顶部生成并动态下滑，慢慢接近玩家所部署的道具建筑物。

(5)实现外星人攻击建筑物时的冒烟效果

外星人会使用冲击波攻击建筑物，建筑物在被攻击后会生成冒烟效果。

(6)实现子弹发射击打外星人效果

玩家在部署基本建筑物后，具有攻击功能的建筑物/道具会发射子弹击打外星人。

(7)实现外星人的攻击冲击波

外星人攻击式会生成攻击冲击波。

(8)通关排行榜模块功能及评论模块功能

系统将不同用户产生的游戏数据进行排序并生成排行榜，用户同时也可以对本游戏在评论模块处进行评价

## 4.3 系统数据库设计

### 4.3.1 E-R模型

注册登录

图4-3 注册登录实体E-R图

评论反馈

图4-4评论反馈实体E-R图

### 4.3.2 数据库表实现

|  |  |  |  |
| --- | --- | --- | --- |
| name | identify | subject | message |
| NULL | NULL | NULL | NULL |

表4-1 Addtest表

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Name | Password | Role |
| 1 | Xyh | 123 | 1 |

表4-2 test表

# 5系统开发与实现

## 5.1 登录界面

5.1.1验证码随机生成功能代码:

// 随机数生成类

Random random = new Random();

// 定义验证码的位数

int size = 5;

// 定义变量保存生成的验证码

String vCode = "";

char c;// 产生验证码

for (int i = 0; i < size; i++) {

// 产生一个26以内的随机整数

int number = random.nextInt(26);

// 如果生成的是偶数，则随机生成一个数字

if (number % 2 == 0) {

c = (char) ('0' + (char) ((int) (Math.random() \* 10)));

// 如果生成的是奇数，则随机生成一个字母

} else {c = (char) ((char) ((int) (Math.random() \* 26)) + 'A');

｝

vCode = vCode + c;}

// 保存生成的5位验证码

request.getSession().setAttribute("vCode", vCode);

// 验证码图片的生成

// 定义图片的宽度和高度

int width = (int) Math.ceil(size \* 20);

int height = 50;

BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

// 获取图片的上下文

Graphics gr = image.getGraphics();

// 设定图片背景颜色

gr.setColor(Color.WHITE);

gr.fillRect(0, 0, width, height);

// 设定图片边框

gr.setColor(Color.GRAY);

gr.drawRect(0, 0, width - 1, height - 1);

// 画十条干扰线

for (int i = 0; i < 5; i++) {

int x1 = random.nextInt(width);

int y1 = random.nextInt(height);

int x2 = random.nextInt(width);

int y2 = random.nextInt(height);

gr.setColor(randomColor());

gr.drawLine(x1, y1, x2, y2);

}

// 设置字体，画验证码

gr.setColor(randomColor());

gr.setFont(randomFont());

gr.drawString(vCode, 10, 22);

// 图像生效

gr.dispose();

// 输出到页面

ImageIO.write(image, "JPEG", response.getOutputStream());

}

// 生成随机的颜色

private Color randomColor() {

int red = r.nextInt(150);

int green = r.nextInt(150);

int blue = r.nextInt(150);

return new Color(red, green, blue);

}

private String[] fontNames = { "宋体", "华文楷体", "黑体", "微软雅黑", "楷体\_GB2312" };

private Random r = new Random();

// 生成随机的字体

private Font randomFont() {

int index = r.nextInt(fontNames.length);

String fontName = fontNames[index];// 生成随机的字体名称

int style = r.nextInt(4);

int size = r.nextInt(3) + 24; // 生成随机字号, 24 ~ 28

return new Font(fontName, style, size);

}

}

### 5.1.2用户名及密码验证功能：

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

request.setCharacterEncoding("utf-8");

response.setContentType("text/html;charset=utf-8");

String userName = request.getParameter("userName");

String password = request.getParameter("password");

System.err.println(userName + ";" + password);

String myUser = "xyh";

String myPwd = "123";

String vericode = request.getParameter("checkcode");

String generatedCode = (String) request.getSession().getAttribute("vCode");

if (userName.equals(myUser) && password.equals(myPwd)

&& vericode.toLowerCase().equals(generatedCode.toLowerCase())) {

response.sendRedirect(request.getContextPath() + "/hello.jsp?userName=" + userName);

} else {

request.setAttribute("message", "账密错误，请重新登录<br>");

request.getRequestDispatcher("/login.jsp").forward(request, response);

}

}

}

## 5.2 游戏功能具体实现代码

### 5.2.1实现物体的拖拽动态特效及创建道具函数：

//创建道具图像

building () {  
        var b = new this.cjs.Container()  
        b.cost = 0  
        return b  
      },  
  
      satellite () {  
        var b = this.building()  
        b.addChild(new this.assetsLib.Satellite())  
        b.cache(-50, -50, 100, 100)  
        b.hp = 150  
        b.cost = 30  
        return b  
      },  
  
      satellite2 () {  
        var b = this.building()  
        b.addChild(new this.assetsLib.Satellite2())  
        b.cache(-50, -50, 100, 100)  
        b.hp = 150  
        b.cost = 600  
        return b  
      },  
  
      castle () {  
        var b = this.building()  
        b.addChild(new this.assetsLib.Castle())  
        b.hp = 300  
        b.shield = 5  
        b.damageDeal = 2  
        b.attackSpeed = 120  
  
        return b  
      },  
  
      castle2 () {  
        var b = this.building()  
        b.addChild(new this.assetsLib.Castle2())  
        b.cache(-50, -50, 100, 100)  
        b.hp = 300  
        b.shield = 10  
        b.damageDeal = 10  
        b.attackSpeed = 200  
  
        return b  
      },  
  
      spaceJunk () {  
        var b = this.building()  
        b.addChild(new this.assetsLib.SpaceJunk())  
        b.hp = 60  
        b.cost = 5  
        return b  
      },

### 5.2.2实现外星人的动态下滑特效：

//创建外星人的类对象

enemy () {  
        var enemyObj = new this.cjs.Container()  
        enemyObj.originalSpeed = 0.5  
        enemyObj.deceleration = 0.4  
        enemyObj.hp = 10  
        enemyObj.damageDeal = 10  
        enemyObj.attackSpeed = 100  
        enemyObj.speed = enemyObj.originalSpeed  
        return enemyObj },

init () {  
this.boardLayer = this.Board()  
this.stage.addChild(this.boardLayer)  
// change here// 手动添加两个敌人对象  
this.addEnemy(&apos;enemyDummy&apos;)  
this.addEnemy(&apos;boss&apos;)  
this.lives = 2},  
tick () {// change here  
 this.boardTick()}// change here  
      boardTick () {// 从敌人对象数组中取出敌人对象，根据属性更新敌人对象在页面上的位置  
        this.enemyMap = this.create2DArray(this.boardLayer.cols, this.boardLayer.rows)  
        for (var i = 0; i < this.enemyList.length; i++) {  
          var enemy = this.enemyList[i]  
          var rowCol = this.screenToRowCol(enemy.x, enemy.y)  
          this.enemyMap[rowCol.col][rowCol.row] = enemy  
          enemy.col = rowCol.col  
          enemy.row = rowCol.row  
          this.enemyTick(enemy)}  
        var len = 0  
        // 当敌人对象落入到页面底部以后将其从页面上删除  
        for (i = 0, len = this.enemyMap.length; i < len; i++) {  
          if (this.enemyMap[i][this.boardLayer.rows] !== undefined) {  
            var bottomEnemy = this.enemyMap[i][this.boardLayer.rows]  
            this.lives -= 1  
            this.removeItem(this.enemyList, bottomEnemy)  
            if (bottomEnemy.parent) {  
              bottomEnemy.parent.removeChild(bottomEnemy)}}}}；

   addEnemy (enemyClass) {  
        var sprite = null  
        switch (enemyClass) {  
          case &apos;enemyDummy&apos;:  
            sprite = this.enemyDummy()  
            break  
          case &apos;boss&apos;:  
            sprite = this.boss()  
            break }  
        this.boardLayer.addChild(sprite)  
        var col = Math.floor(Math.random() \* this.boardLayer.cols)  
        var pos = this.rowColToScreen(0, col)  
        sprite.x = pos.x  
        sprite.y = pos.y  
        sprite.row = 0  
        sprite.col = col  
        this.enemyList.push(sprite)},

### 5.2.3气泡的螺旋升腾特效

energy (x, y) {  
        var obj = new this.cjs.Container()  
        obj.addChild(new this.assetsLib.Energy())  
        obj.cache(-25, -25, 50, 50)  
        obj.x = x || 0  
        obj.y = y || 0

// 记录起始x坐标  
        obj.baseX = obj.x  
        // 膨胀1.2倍再缩回原有尺寸  
        this.cjs.Tween.get(obj, {loop: true}).to({scaleX: 1.2, scaleY: 1.2}, 600).to({scaleX: 1.0, scaleY: 1.0}, 600)  
        obj.on(&apos;mousedown&apos;, this.energyClick)  
        this.energyList.push(obj)  
        console.log(&apos;create energy: &apos;, obj)  
        return obj},  
energyClick (e) {  
        console.log(&apos;energyClick: &apos;, e.currentTarget)  
        this.energies += 100  
        this.removeItem(this.energyList, e.currentTarget)  
        e.currentTarget.parent.removeChild(e.currentTarget)  
      }

### 5.2.4实现外星人攻击建筑物时的冒烟效果

building () {  
        var b = new this.cjs.Container()  
        b.cost = 0  
        // change here  
        b.shield = 10  
        b.damageDeal = 0  
        b.attackSpeed = 9999 return b},  
buildingDamage (building, damage) {  
        var realDamage = Math.max(damage, building.shield, 0)  
        building.hp -= realDamage  
        if (building.hp <= 0 && building.parent) {  
          this.removeBuilding(building)  
removeBuilding (building) {  
     this.boardLayer.buildingMap[building.col][building.row] = undefined  
     this.boardLayer.removeChild(building)  
 },enemyAttackBuilding () {// 当外星人与道具相遇时，攻击道具  
        for (var i = this.enemyList.length - 1; i >= 0; i--) {  
          var enemy = this.enemyList[i]  
          var row = enemy.row  
          var col = enemy.colvar target  
          if (this.boardLayer.buildingMap[col][row] !== undefined) {  
            target = this.boardLayer.buildingMap[col][row]  
          } else if (this.boardLayer.buildingMap[col][row + 1] !== undefined) {  
            target = this.boardLayer.buildingMap[col][row]}  
          if (target !== undefined) {  
            enemy.speed -= enemy.deceleration  
            this.enemyStartAttack(enemy, target)  
          } else {  
            this.enemyStopAttack(enemy)}  
          target = undefined}},

enemyStartAttack (enemy, target) {  
        if (!enemy.isAttacking) {  
          enemy.isAttacking = true  
          enemy.addChild(enemy.attackingSmoke)  }  
        enemy.attackingTarget = target }, // change here  
      enemyStopAttack (enemy) {  
        if (enemy.isAttacking) {  
          enemy.removeChild(enemy.attackingSmoke)}  
        enemy.isAttacking = false  
        enemy.attackingTarget = undefined  
        enemy.speed = enemy.originalSpeed},

5.2.5实现外星人的攻击冲击波

// change 6  
      waveTick () {  
        if (!this.waves.isActive) {  
          return }  
        if (this.cjs.Ticker.getTicks() % this.waves.currentWave.frequency === 0) {  
          // 依次把不同种类的外星人召唤出来  
          var accumunateTargetCount = 0  
          for (var i = 0, len = this.waves.enemySummonOrder.length; i < len; i++){  
            var enemyType = this.waves.enemySummonOrder[i] // 获得外星人个数  
            var targetCount = this.waves.currentWave[enemyType] || 0  
            accumunateTargetCount += targetCount  
            if (this.waves.enemiesSummoned < accumunateTargetCount) {  
              brea }}  
          if (this.waves.enemiesSummoned >= accumunateTargetCount) {  
            this.waves.isActive = false  
          } else {  
            this.addEnemy(enemyType)  
            this.waves.enemiesSummoned += 1}}},  
      tick () { // change 9  
        this.wavesSpan.textContent = this.waves.nextWave + 1  
        this.boardTick()// change 7  
        this.waveTick()  
        if (!this.waves.isActive && this.areEnemiesCleared()) {  
          this.boardLayer.isAddingBuilding = false  
          this.removeAllBuildings()  
          this.removeAllBullets()  
          this.wavesCleared()// 显示字体动画  
          this.cjs.Tween.get(this.nextWaveSprite).to({x: this.canvas.width / 2, y: this.canvas.height / 2, alpha: 0}).to({alpha: 1.0}, 300).wait(1000).to({alpha: 0}, 300).to({x: 999}) this.startWave()}} // change 8  
      removeAllBuildings () {  
        for (var i = 0; i < this.boardLayer.cols; i++) {  
          for (var j = 0; j < this.boardLayer.rows; j++) {  
            this.removeBuilding(this.boardLayer.buildingMap[i][j])} }  
        this.satelliteList = []  
        this.castleList = []},  
      areEnemiesCleared () {  
        return this.enemyList.length === 0 },  
      removeAllBullets () {  
        this.bulletList.length = 0  
        this.effectLayer.removeAllChildren()  
      },

# 6系统测试

## 6.1 首页

首页1:web页面实现功能说明，首页三个界面自动轮播，也可直接点击上方导航栏跳转到其他页面

首页2：项目介绍界面

介绍了本项目的框架以及小组成员和项目名称，用户也可点击按钮跳转到登录页面

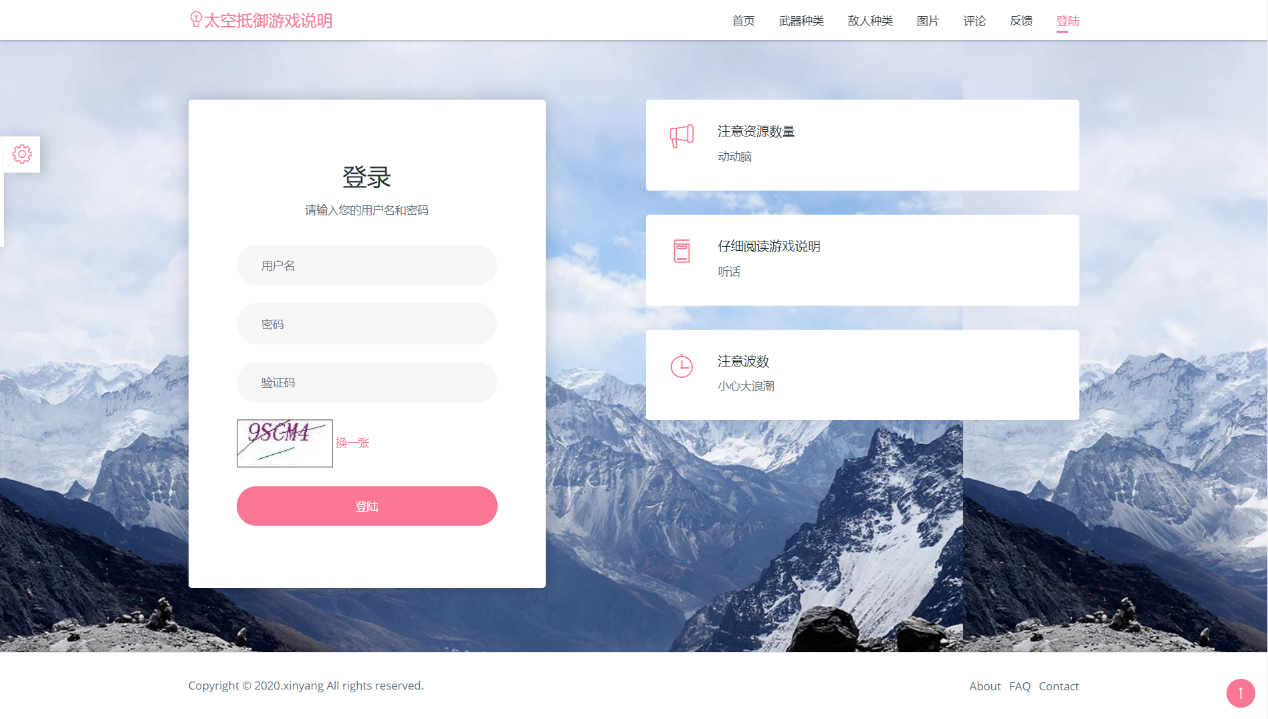
首页3：

描述了项目使用的语言，简单介绍了本游戏的玩法，点击按钮即可跳转到登录界面



## 6.2 登录界面

用户可以在此进行登录，并且点击右上方进入武器种类，敌人种类，图片，评论，反馈界面。如若输入错误的用户名和密码或验证码，则刷新页面，用户需重新输入用户名密码及验证码。右侧为简单的游戏小提示，有一定的参考价值，用户可仔细阅读。



## 6.3 功能说明界面

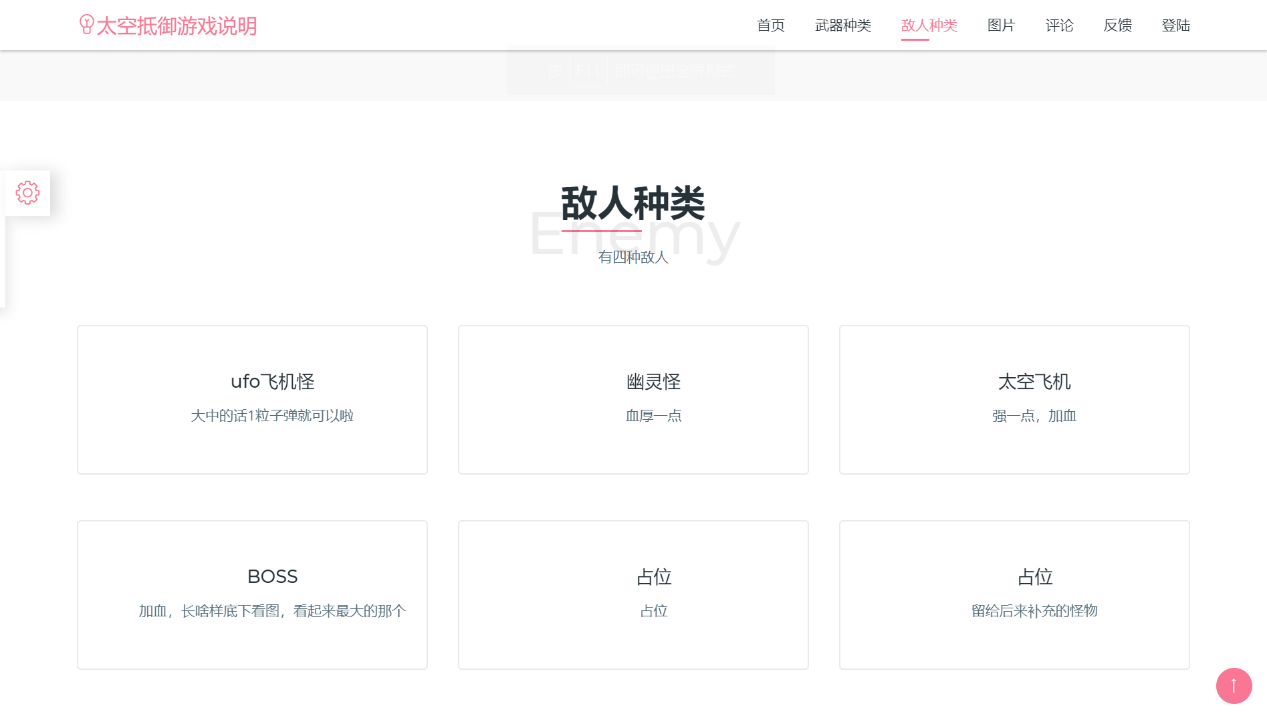
### 6.3.1 武器种类

描述了武器种类和游戏设定介绍，武器种类有：太空垃圾类，卫星类，卫星强化版，炮台，炮台强化班，以及新的待定武器（待做）

用户可以在此界面了解游戏的设定，并了解每个武器的种类和功能以及如何使用，由此对游戏进行部署，增强了游戏的可玩性。

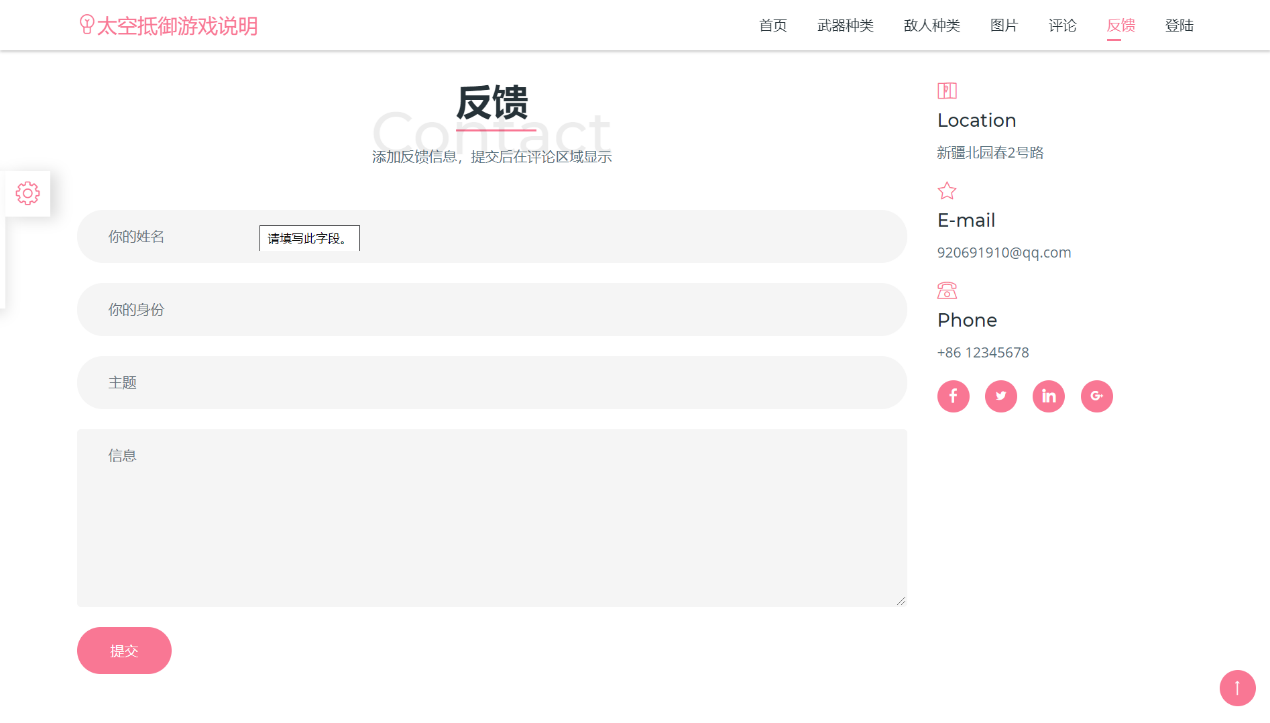


### 6.3.2 敌人种类

用户可以在此得知游戏的敌人种类：ufo飞机怪，幽灵怪，太空飞机，boss，以及待补充的两个种类。用户可以由此得知敌人的特征以及击败方法，从而合理地部署自己的武器以取得更好的游戏体验。

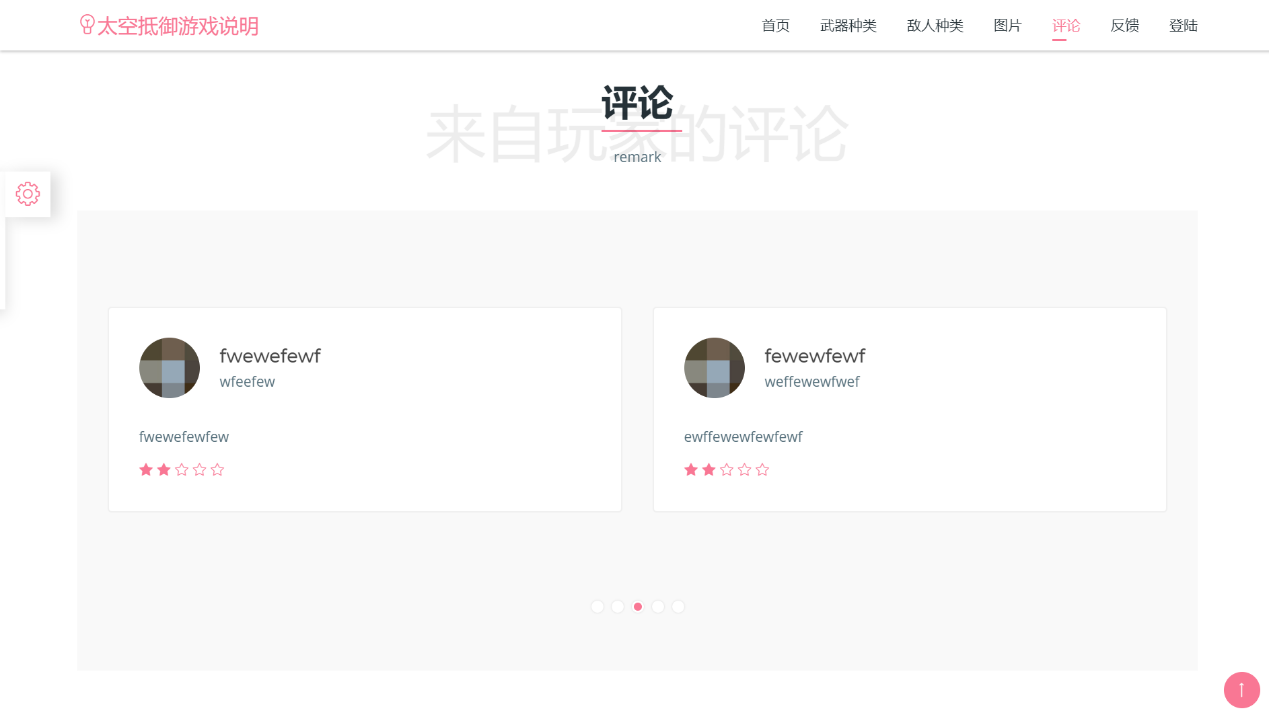
### 6.3.3 反馈界面

用户在此输入姓名，身份，主题及信息后，可对本游戏进行反馈，添加反馈信息成功后，将在评论区显示用户的反馈内容。用户同时可根据右侧的联系方式进行反馈。

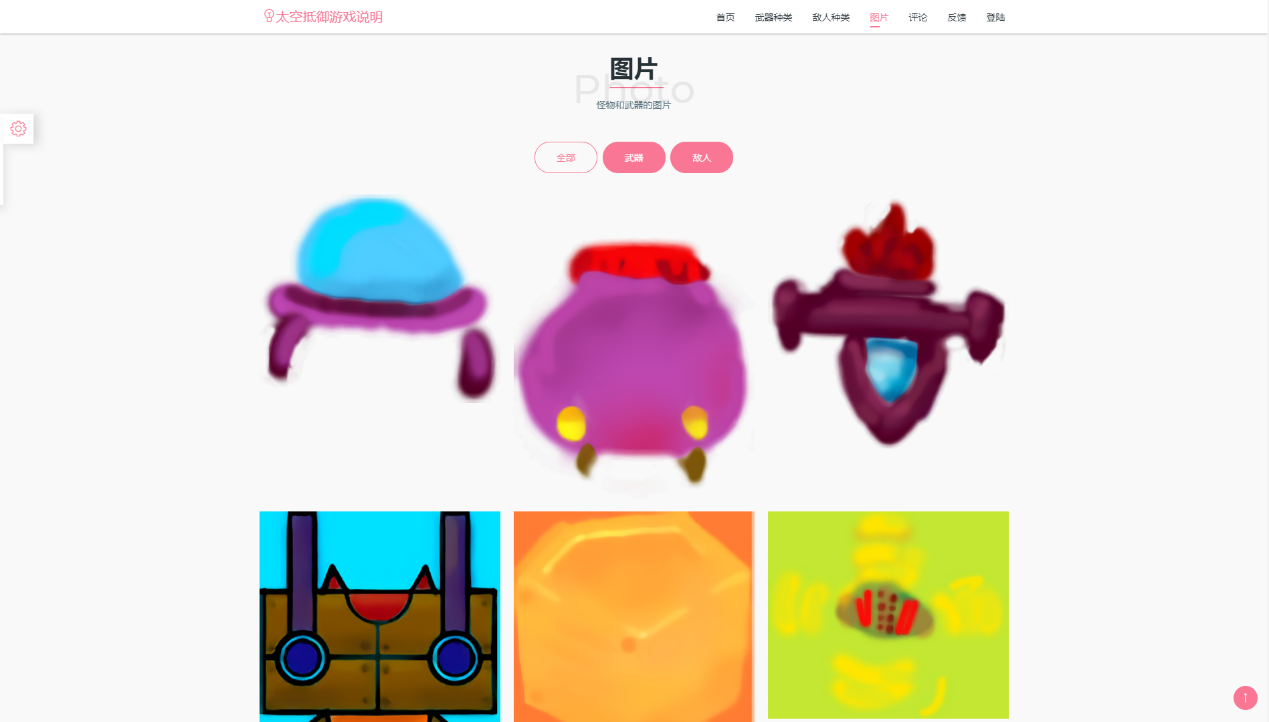


### 6.3.4 评论界面

在反馈信息成功录入数据库后，用户所添加的反馈信息将会显示在评论区，其他用户可根据他人的反馈信息来判断本游戏的可玩性，同时也可在评论区进行游戏玩法交流。



### 6.3.5 图片界面

用户可以在此界面看到敌人的具体模样，不同的敌人有不同的形态，同样，不同的道具也有不同的形态，本界面可以为用户熟悉游戏的敌人与道具提供良好的接口。

### 6.3.6 游戏界面

右侧显示用户的id，账号，以及获得的分数。玩家可以上传自己的分数至游戏排行榜上，也可点击按钮查看自己此时的分数。



# 7 总结与展望

## 7.1总结

在这次课程设计中，我基本上完成了团队分配的任务任务，实现了在JAVAWEB基础上的基础界面设计,基本上达到了此次课程设计的要求。

虽然在这次的课程设计中有很多的不足，但是我也有很多的收获。这让我不仅对JAVAWEB开发环境有了更多的认识和了解，而且也使我对MYSQL数据库有了更加深入具体的了解，让我的理论知识和算法与数据库的使用有了更多的实践。作为一个独立完成的项目，我更体会到了自己学到的知识的重要性,深刻地认识到只有努力地学习并研究相关的知识，才能提高做事的效率，也才能更好更快地完成任务。所以这将在我以后的学习和生活中有很在的引导作用。在以后的学习中，我也将加倍地学习编程的有关知识和数据库的有关知识，以同样的方法学习其他的知识，不断地丰富自己，充实自己，提高自己的能力。

## 7.2展望

基于WEB的前段页游的设计与实现涉及多方面的理论、方法和技术，本系统还有许多新的问题需要解决，需要在实际应用中不断积累和完善，在以下几个方面，还需要做进一步的研究和开发。  
  1.论文只考虑了游戏内容道具的几项基本组成，并未在可玩性设计等方面进行深入发掘。使得系统只考虑了一些简单情况，如何应付用户的多种道具的选择还值得加强。可以进一步完善游戏道具，规则设置的相关字段，增加游戏信息的完整性。  
  2.该系统只考虑了玩家的一些基本情况，未对评论及排行榜等扩展功能进行更为深入的研究，使得游戏部署的一些接口还不够完善。  
  3.系统在测试上还存在一些问题，不过都是预期可以解决的。  
 相关WEB技术在该系统的开发过程中应用范围还不够，还有提高的余地。

# 参考文献

[1] 俞东进、任祖杰编著 Java EE Web 应用开发基础 电子工业出版社 2012年

[2] 张跃平、耿祥义编著 JSP程序设计 清华大学出版社 2009年

[3] 许令波编著 深入分析Java Web技术内幕 电子工业出版社 2012年