**软件工程第一小组用户手册**

指导老师：杨磊

组员：孙威 叶俊岑 江林桓 李筱 张伟强

注：本手册为软件第一版手册，仅供使用者们借鉴参考

**目录**

1. **引言**
2. **软件概述**
3. **运行环境**
4. **使用说明**
5. **运行说明**
6. **非常规过程**
7. **操作命令一览表**
8. **程序文件和文件一览表**

**引言**

**1.1编写的目的：**

为了进一步让使用者（广大青年以及相关技术前辈）认识了解到我们游戏软件的相关情况，运行安全性，各个操作命令的相关内容以及游戏软件的相关构架，特编写游戏软件的用户手册来向用户们展示我们软件的情况‘

**1.2项目的背景**：

现代生活节奏的加快，很少人会停下脚步去品味一下慢节奏的生活。手机已经成为人们日常生活中一种必不可少的工具，电脑的许多功能逐步在手机上得以实现开发一款游戏用于吸引广大青年，将电脑上的游戏搬到手机中去，这是我们编写此游戏软件的初始目的，同时也是对团队软件工程建设能力的一种锻炼。本次游戏软件是基于电脑小游戏——僵尸危机3的一款改装小游戏。’用智能手机游戏抢占青少年市场，软件等程序的开发源于小组中代码能力较强者。

**1.3参考资料：C++面向对象程序设计（第二版） 谭浩强 编著 清华大学出版社**

**构建之法现代软件工程 邹新 著 人民邮电出版社**

**软件概述**

**2.1目标**

通过各种系统板块方面的完善能够提升软件的完整性，通过增加板块扩展游戏本身的内容，进一步吸引广大游戏玩家，同时联系广告商和项目合作团队打造更适合的智能手机版的电脑小游戏

**2.2功能**

本游戏软件是手机版本而运行的，通过手机客户端下载安装游戏成功后可进入游戏，游戏难度分为：简单，普通，困难三种，使用者可以根据实际情况选择适当选择关卡难度同时可以根据自己的喜好调节游戏时的亮度和音量，角色通过武器的选择与僵尸作战来达到娱乐的目的。此款软件有一个可以恢复数据的功能，当在进入游戏时有一个读取游戏的选项，通过点击此选项可恢复上次游戏的进度继续游戏，第一次进入游戏除外。帮助选项可以让游戏使用者了解到如何进行游戏

**2.3性能**

在数据精确度时间特性方面方面此款软件具有较高的可靠性，软件设计的初始版本源自电脑游戏，当将其版本改为手版本后，经过调试和用户体验后，在反馈信息中体验用户对本游戏软件具有较高的认可度。由于本软件是为手机用户而开发的在灵活性方面比电脑版操作灵活人物角色反应时间大大缩短平均人物在每秒内能够完成4-5个动作，在软件运作方面已经在模拟器上能够完美展现，下方为游戏运行时的截图，仅供参考

****

**运行环境**

本款软件是面向于智能手机用户，对于现在市面上流行的智能手机来说均可以运行本游戏支持能使用4G网的智能手机，同时应具备较强的信息处理能力游戏时要保持网络的畅通以保证游戏按键与人物角色的行动的匹配。

在游戏过程中会出现以下情况：在校园网链接的情况下游戏体验会与正常情况下有些不同，具体表现为游戏运行延迟，玩家操作后角色人物无法及时完成相应的从步骤，在正常网络连接的情况下玩家可以向系统购买加速器，来减少人物角色延迟现象（注加速器在校园网链接的情况下游戏加速会受到限制约是正常情况下使用加速器的80%、例如在非校园网连接的情况下网络游戏的模块读取速度为10，通过加速器可到20，在校园网连接的情况下模块读取速度在7-8之间通过加速器可达到15,。）

**使用说明**

**4.1程序的安装和初始化**

本款软件是以安装包的的形态供用户下载，当用户从网上找到本游戏的资源时通过智能手机的专用下载软件（如vivo的应用商店，应用宝等等软件下载工具）下载本游戏包裹。本款游戏现阶段大小为25mb左右占手机的系统内存少当软件下载完毕后可自行安装于手机内存，安装完成后即可进入游戏界面。体验本游戏的乐趣。

**4.2相关操作后游戏内部反应对照**

在开始菜单：

1.开始游戏：开始游戏： 游戏界面 人物死亡后 结束界面（提供返回到开始菜单的选项）。

2.设置：设置 游戏难度 简单 普通 困难 返回按钮

音量 调节音量大小 返回按钮

场景亮度 调节亮度 返回按钮

3.帮助：帮助文档 显示帮助文档 返回按钮

4.读取游戏：读取游戏 读取上一次游戏进度。

**4.3用户的反馈方式以及问题的解决方法**

在游戏过程中一定会出现所谓的bug，对于用户的体验我们十分看重。当游戏出现bug后用户可以通过官方渠道向我们提出，对于向我们提出宝贵建议的用户我们会对其账号给予一定的奖励并对用户所提的bug进行修复，奖励会在最近期维护后发放。例如：用户在游戏过程中发现在将要杀死僵尸boss时游戏突然提示系统运行异常，并强行下线，在次上线后游戏数据丢失。当用户向我们反映后我们会及时调取服务器中相应时间段的数据情况，经查证情况属于漏洞时，我们一边在做出修改的同时会向该用户提供一定的奖励。本软件的建议反馈通道将会全天开放，随时随地接受用户的反馈。

**运行说明**

**游戏过程中相关显示信息**

1.信息显示：在游戏画面中规划出一块区域用于显示此时角色的血量，所使用枪支的名字，子弹的剩余量，关卡的名字。并会随着游戏的进行发生改变。

2.枪支系统：按下切换枪支按钮，切换不同的枪。

3.积分系统：在后台运行，记录下玩家的分数，在游戏结束后显示。

4.碰撞检测系统：后台时刻记录下僵尸，角色，子弹，地图的位置及关系，并通过各种判断选择调用何种函数。

**非常规过程**

当在游戏过程中出现卡死，游戏界面闪烁，角色人物的实际行动与玩家操作不匹配或其他不同情况时可直接按游戏界面的返回键，维护人员在维护时要根据具体出现的情况在合适的板块维护。

**维护板块及对应的程序板块**

1. 信息显示失败——信息显示模块
2. 武器切换失败——板块链接模块
3. 武器子弹与武器实际不符——武器函数模块

**操作命令一览表**

1.操作：通过虚拟摇杆来控制角色行动的方向，开枪的方向，通过虚拟的按键控制角色开枪和切换枪支。

2.暂停：按下暂停按钮 游戏画面停止，按钮变为启动形状 再次点击

游戏画面继续。

**程序文件和数据文件**

**一下为程序部分分别对应相关游戏内的各个部件**

**Bullet 子弹程序**

**Enemy 僵尸生成程序**

**Help 帮助选项**

**Loading 等待程序**

**Main act 主要动作程序**

**Menu 菜单页面程序**

**Boom,game surface,option,role,start pause,total menu等各个程序模块的程序**

**接下来主要介绍**

**Bullet程序**

package com.example.finalproject;

import android.graphics.Bitmap;

import android.graphics.Canvas;

import android.graphics.Paint;

public class Bullet implements Runnable{

int x,y;

Bitmap bmpBulletup;

Bitmap bmpBulletdown;

Bitmap bmpBulletleft;

Bitmap bmpBulletright;

Boolean life=true;

int direction;

int speed=5;

int harm=1;

public Bullet(Bitmap bmpBulletup,Bitmap bmpBulletdown,Bitmap bmpBulletleft,Bitmap bmpBulletright,int x,int y,int direction){

this.bmpBulletup=bmpBulletup;

this.bmpBulletdown=bmpBulletdown;

this.bmpBulletleft=bmpBulletleft;

this.bmpBulletright=bmpBulletright;

this.x=x;

this.y=y;

this.direction=direction;

}

public void draw(Canvas canvas,Paint paint){

if(this.direction==1){

canvas.drawBitmap(this.bmpBulletup, this.x,this.y, paint);

}

if(this.direction==2){

canvas.drawBitmap(this.bmpBulletdown, this.x,this.y, paint);

}

if(this.direction==3){

canvas.drawBitmap(this.bmpBulletleft, this.x,this.y, paint);

}

if(this.direction==4){

canvas.drawBitmap(this.bmpBulletright, this.x,this.y, paint);

}

}

//线程函数

@Override

public void run() {

// TODO Auto-generated method stub

while(true){

try {

this.logic();

Thread.sleep(100);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

public void logic(){

if(this.direction==1){

this.y-=this.speed;

if(x+bmpBulletup.getWidth()<0||x>GameSurfaceView.screenw||y+bmpBulletup.getHeight()<0||y>GameSurfaceView.screenh){

this.life=false;

}

}

if(this.direction==2){

this.y+=this.speed;

if(x+bmpBulletdown.getWidth()<0||x>GameSurfaceView.screenw||y+bmpBulletdown.getHeight()<0||y>GameSurfaceView.screenh){

this.life=false;

}

}

if(this.direction==3){

this.x-=this.speed;

if(x+bmpBulletleft.getWidth()<0||x>GameSurfaceView.screenw||y+bmpBulletleft.getHeight()<0||y>GameSurfaceView.screenh){

this.life=false;

}

}

if(this.direction==4){

this.x+=this.speed;

if(x+bmpBulletright.getWidth()<0||x>GameSurfaceView.screenw||y+bmpBulletright.getHeight()<0||y>GameSurfaceView.screenh){

this.life=false;

}

}

}

//造成伤害

public void doharm(Enemys e){

e.life-=harm;

}

}

//手枪子弹类

class Piston extends Bullet implements Runnable{

public Piston(Bitmap bmpBulletup,Bitmap bmpBulletdown,Bitmap bmpBulletleft,Bitmap bmpBulletright,int x, int y,int direction) {

super(bmpBulletup,bmpBulletdown,bmpBulletleft,bmpBulletright, x, y,direction);

// TODO Auto-generated constructor stub

this.harm=1;

this.speed=15;

}

public void draw(Canvas canvas,Paint paint){

if(this.direction==1){

canvas.drawBitmap(this.bmpBulletup, this.x,this.y, paint);

}

if(this.direction==2){

canvas.drawBitmap(this.bmpBulletdown, this.x,this.y, paint);

}

if(this.direction==3){

canvas.drawBitmap(this.bmpBulletleft, this.x,this.y, paint);

}

if(this.direction==4){

canvas.drawBitmap(this.bmpBulletright, this.x,this.y, paint);

}

}

}

//散弹子弹类

class Shotgun extends Bullet implements Runnable{

public Shotgun(Bitmap bmpBulletup,Bitmap bmpBulletdown,Bitmap bmpBulletleft,Bitmap bmpBulletright,int x, int y,int direction){

super(bmpBulletup,bmpBulletdown,bmpBulletleft,bmpBulletright, x, y,direction);

// TODO Auto-generated constructor stub

this.harm=3;

this.speed=15;

}

public void draw(Canvas canvas,Paint paint){

if(this.direction==1){

canvas.drawBitmap(this.bmpBulletup, this.x,this.y, paint);

}

if(this.direction==2){

canvas.drawBitmap(this.bmpBulletdown, this.x,this.y, paint);

}

if(this.direction==3){

canvas.drawBitmap(this.bmpBulletleft, this.x,this.y, paint);

}

if(this.direction==4){

canvas.drawBitmap(this.bmpBulletright, this.x,this.y, paint);

}

}

}

//地雷类

class mine extends Bullet implements Runnable{

public mine(Bitmap bmpBulletup, Bitmap bmpBulletdown, Bitmap bmpBulletleft, Bitmap bmpBulletright, int x, int y,

int direction) {

super(bmpBulletup, bmpBulletdown, bmpBulletleft, bmpBulletright, x, y, direction);

// TODO Auto-generated constructor stub

this.harm=5;

this.speed=0;

}

public void draw(Canvas canvas,Paint paint){

if(this.direction==1){

canvas.drawBitmap(this.bmpBulletup, this.x,this.y, paint);

}

if(this.direction==2){

canvas.drawBitmap(this.bmpBulletdown, this.x,this.y, paint);

}

if(this.direction==3){

canvas.drawBitmap(this.bmpBulletleft, this.x,this.y, paint);

}

if(this.direction==4){

canvas.drawBitmap(this.bmpBulletright, this.x,this.y, paint);

}

}

}