飞机大战开发文档

## 一、项目整体介绍

### 1、项目描述

飞机大战是一款经典的射击游戏，玩家需要控制自己的飞机，躲避敌机的攻击并尽可能地消灭它们。本项目基于Felgo框架开发，具有良好的跨平台性，支持iOS、Android、Windows等多个平台。本项目采用了最新的OpenGL ES 2.0技术，实现了高质量的游戏画面和流畅的游戏体验。玩家可以通过触屏或键盘操控飞机，射击敌机，收集道具，升级武器等。

游戏中有多个关卡，每个关卡都有不同的敌机和道具，玩家需要不断提升自己的技能和装备，才能挑战更难的关卡。此外，游戏还提供了多种游戏模式，如生存模式、挑战模式等，让玩家可以根据自己的喜好选择不同的游戏体验。

### 2、项目背景

暂无

## 二、研发任务分配

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **任务1** | | | | | |
| 开始时间： | | 2023.6.12 上午 10：30 | | 结束时间： | 2023.6.12 晚上 10：30 |
| **任务说明及分配** | | | | | |
| **编号** | **执行者** | | **任务详细说明** | | |
| 1 | 王 | | 分配任务，构建游戏程序初始类图，尝试熟悉felgo程序 | | |
| 2 | 张 | | 制作开发文档，尝试熟悉felgo程序，收集素材 | | |
| 3 | 邹 | | 熟悉felgo程序逻辑，以及飞机大战游戏模型 | | |
| **任务成果** | | | | | |
| all | 基本上全部完成，其中素材较少 | | | | |
| **未完成任务** | | | | | |
| **编号** | **详细原因分析** | | | | |
| 2 | 素材不太好找 | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **任务2** | | | | | |
| 开始时间： | | 2023.6.13 上午 8：30 | | 结束时间： | 2023.6.15 晚上 10：30 |
| **任务说明及分配** | | | | | |
| **编号** | **执行者** | | **任务详细说明** | | |
| 1 | 王 | | 熟悉felgo语法，完善初始类图 | | |
| 2 | 张 | | 熟悉felgo语法，整理资源，及时完善项目文档 | | |
| 3 | 邹 | | 熟悉felgo语法，尝试重构初始版本飞机大战 | | |
| **任务成果** | | | | | |
| **编号** |  | | | | |
|  | 基本完成 | | | | |
| **未完成任务** | | | | | |
| **编号** | **详细原因分析** | | | | |
| 3 | 基础版本飞机大战以不适合当下情形，已经重新分配任务 | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **任务3** | | | | | |
| 开始时间： | | 2023.6.16 上午 10：30 | | 结束时间： | 2023.6.18 晚上 10：30 |
| **任务说明及分配** | | | | | |
| **编号** | **执行者** | | **任务详细说明** | | |
| 1 | 王 | | 分析并初始编写己方飞机类 | | |
| 2 | 邹 | | 分析并初始编写菜单和游戏背景 | | |
| 3 | 张 | | 分析并初始编写敌机类 | | |
| **任务成果** | | | | | |
| **编号** |  | | | | |
| all | 基本完成 | | | | |
| **未完成任务** | | | | | |
| **编号** | **详细原因分析** | | | | |
|  |  | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **任务4** | | | | | |
| 开始时间： | | 2023.6.19 上午 10：30 | | 结束时间： | 2023.6.20 晚上 10：30 |
| **任务说明及分配** | | | | | |
| **编号** | **执行者** | | **任务详细说明** | | |
| 1 | 王 | | 分析并编写敌机运动逻辑 | | |
| 2 | 邹 | | 分析并编写启动界面动画和设置界面 | | |
| 3 | 张 | | 分析并初始编写道具运动逻辑 | | |
| **任务成果** | | | | | |
| **编号** |  | | | | |
| all | 基本完成 | | | | |
| **未完成任务** | | | | | |
| **编号** | **详细原因分析** | | | | |
|  |  | | | | |

## 三、问题分析处理

### 现阶段出现的问题

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 发现时间 | 问题描述 | 解决方案 |
| 1 | 6.12 17:30 | 合适游戏素材难找 | 暂无 |
| 2 | 6.14 16:30 | 控制飞机射击无法向左边 | 暂无 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### 2、可能出现的问题

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 发现时间 | 问题描述 | 解决方案 |
| 1 | 6.15 09:10 | 飞机移动不流畅 | 使用物理引擎 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 四、项目分析设计

### 1、游戏介绍

本项目是一款基于felgo开发的飞机大战游戏，游戏玩法类似于经典的飞机大战游戏，玩家需要操作自己的飞机，躲避敌机的攻击并消灭它们，最终达到过关的目标。

游戏界面采用了精美的图形和音效，使得游戏更加具有吸引力和可玩性。玩家可以通过不断击败敌机获取积分，解锁更多的飞机和道具，提升自己的战斗能力。

游戏还支持多种游戏模式，包括单人模式、多人对战模式等，让玩家可以与其他玩家进行实时对战，体验更加刺激的游戏体验。

### 2、概要设计

概要设计如下：

1. 游戏简介：飞机大战是一款基于felgo开发的经典飞行射击游戏，玩家需要操作自己的飞机，在不断躲避敌机的攻击和追击中，消灭敌机并最终达成过关目标。
2. 游戏模式：本游戏支持单人模式，单人模式中玩家需要完成一系列任务和挑战
3. 游戏界面：游戏界面采用精美的图形和音效，包括游戏场景、飞机、敌机、道具等，让玩家感受到真实的飞行和战斗体验。
4. 游戏操作：玩家通过手指滑动屏幕或点击垫盘对应按键控制自己的飞机移动和发射子弹，消灭敌机和获得道具。
5. 游戏关卡：游戏设有多个关卡，每个关卡都有不同的地图、敌机和任务，玩家需要在规定时间内完成任务和击败所有敌机才能过关。
6. 游戏道具：游戏中设有多种道具，包括加速、加强攻击、回复生命值等，玩家可以通过消灭敌机或者完成任务获得。
7. 游戏设置：游戏设有设置选项，玩家可以自由设置音效、音乐、语言等。
8. 游戏更新：游戏将会不断更新，添加新的关卡、道具和功能，为玩家带来更多的乐趣和挑战。

#### 2.1、游戏主要功能模块介绍

##### 2.1.1、游戏模块

开发飞机大战游戏需要设计以下游戏类：

1. 飞机类：包括玩家的飞机和敌机，需要设计不同类型的飞机、飞机的移动、攻击等行为。

2. 子弹类：包括玩家和敌机的子弹，需要设计不同类型的子弹、子弹的移动、攻击等行为。

3. 道具类：包括加护盾、加强攻击、回复生命值等道具，需要设计道具的获取、使用和效果。

4. 地图类：包括不同类型的地图、地图中的障碍物、道具等，需要设计地图的生成和加载。

5. 碰撞检测类：用于检测飞机、子弹、道具之间的碰撞，判断是否造成伤害或者获得道具。

6. 游戏控制类：用于控制游戏的开始、暂停、结束、计分等功能。

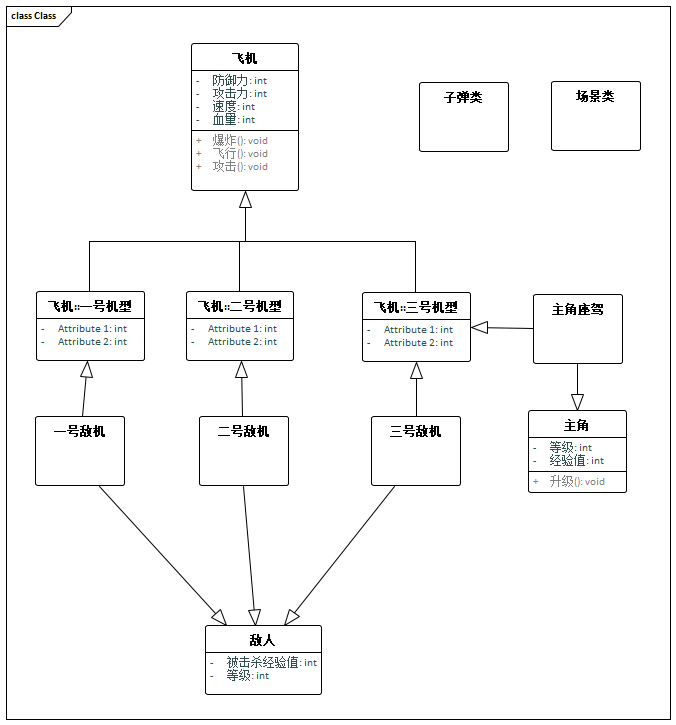
7. 用户数据类：用于存储玩家的游戏数据，包括游戏得分、游戏进度等。

9. 音效类：用于播放游戏中的音效和音乐。

10. 界面类：用于设计游戏的各种界面，包括主菜单、游戏设置等。

#### 2.2、 UML类图

##### 2.2.1、游戏逻辑初始类图：



### 3、详细设计

#### 3.1、介绍每个类的实现

##### 3.1.1、游戏模块

游戏模块分为三部分：界面（UI）模块、玩家（PLayer）模块和敌人（Enemy）模块

#### 3.2、设计思路及采用的先进思想

##### 3.2.1、主人公死亡后闪动动画的实现：

主人公死亡后，屏幕闪过你已死亡的文字弹幕，通过一个矩形框移动配合Text文字模块显示，具体如下：

Rectangle {

id: bossRec

//text: qsTr("游戏失败")

//警告:

color: "red"

//Font.pixelSize: 50

visible: false

//opacity: 0.8

anchors.fill:parent

SequentialAnimation on opacity {

loops: Animation.Infinite

PropertyAnimation {

to: 0

duration: 500 // 1 second for fade out

}

PropertyAnimation {

to: 0.2

duration: 500 // 1 second for fade in

}

}

}

##### 3.2.2、检测前方能否的方法：

通过使用felgo中自带的BoxCollider实现碰撞检测，如果触发条件，则阻止该实体的位置属性的变化，实现墙的效果，集体实现如下：

EntityBase {

id: entity

entityType: "wall"

BoxCollider {

id: boxCollider

bodyType: Body.Static

// the size of the collider is the same as the one from entity by default

}

Rectangle {

anchors.fill: parent

color: "brown"

// this could be set to true for debugging

visible: false

}

}

##### 3.2.3、爆炸检测方法：

爆炸检测与碰撞检测相似，只是修改触发实体为子弹和敌机之间的碰撞，具体实现如下：

BoxCollider {

id: boxCollider

// the image and the physics will use this size; this is important as it specifies the mass of the body! it is in respect to the world size

width: 40

height: 5

anchors.centerIn: parent

density: 0.003

friction: 0.4

restitution: 0.5

body.bullet: true

// we prevent the physics engine from applying rotation to the rocket, because we will do it ourselves

body.fixedRotation: true

fixture.onBeginContact: {

var fixture = other;

var body = other.getBody();

var otherEntity = body.target

//collisionSound.play()

// get the entityType of the colliding entity

var collidingType = otherEntity.entityType

/\*

if(collidingType === "car" ||

collidingType === "rocket") {

entity.removeEntity()

return

}\*/

if(collidingType === "wall") {

entity.removeEntity()

return

}

##### 3.2.4、怪的智能化：

怪的智能化只要体现在怪物的子弹可以有追踪效果，并且怪物可以随着游戏时间推进和积分变化实时更新怪物难度，甚至增加Boss等等，具体如下：

onScoreChanged: {

if(score>=200){

t0.running=false

t1.running=false

t2.running=false

//entityManager.createEntityFromComponent(bossCompent)

boss.visible=true

//bossRec.visible=true

bossComing.visible=true

bossControl.running=true

}

}

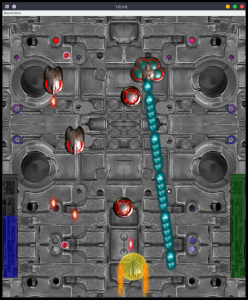
### 4、UI设计

#### 4.1、界面设计

##### 4.1.1、菜单画面：



##### 4.1.2、主程序画面：



#### 4.2、运行平台及测试平台

felo 3.10.0.1 desktop for qt 5.15.2

### 5、参考资料

—felgo官方游戏平台：[game.felgo.com](http://game.felgo.com)

—felgo程序中的官方文档

—CSDN平台中关于felgo程序开发的相关教程