《等风来》

软件设计文档

——等风来小组 2015.7.18

源文件下载：https://github.com/WaitingForWind/WaitingForWind

声明：　　源文件中 AppDelegate.h/.cpp 系由 Cocos2d-x 工程自动提供的，我们仅在里面做少部分的修改。其余的源文件都属原创，使用及转发请注明。

目录

1.介绍

1.1 目的

《等风来——设计文档》给安卓端游戏《等风来》规划出整个游戏的纲要，提供了开发时的详细设计，帮助开发人员了解整个系统的设计。《等风来》是一款安卓端的益智类、休闲类游戏，让人们在无聊的时候用以消遣娱乐。

1.2 范围

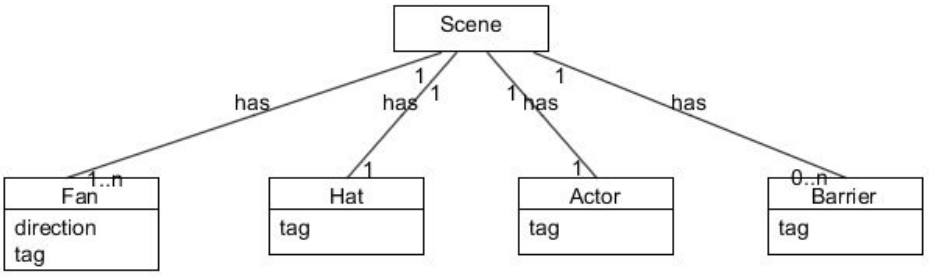
这份文档包含了《等风来》的详细设计。游戏在安卓手机上运行，具有移动性，易上手，具有挑战性的益智通关类游戏。游戏将采用 C++和 Java 编写，在 Android SDK 和 NDK 的联合下，使用 C++驱动 Cocos2d-x 引擎，让 OpenGL ES 在 Android 窗体上运行，从而使游戏流畅运行。

1.3 术语表

术语	描述
C++	面向对象编程语言
C	面向过程语言
Java	有 Sun 公司推出的一种非常方便的面向对象编程语言
OpenGL ES	即 Open Graphic Library（Embed System），开源的计算机图像库，ES 则是专门为嵌入式设备开发的版本。需用 C 语言支持
Cocos2d-x	一款支持跨平台开发的开源游戏引擎，支持 win32，win8，iOS 和 Android 等主流平台。该引擎由 C++编写与使用
Android	由 Google 公司推出的移动设备操作系统
Visual Studio 2012	由微软开发的一款非常强力的开发平台
Object Oriented	面向对象，是一种编程思想，本游戏全程使用面向对象开发

2.设计

2.1 数据模型



- Scene: 指游戏场景
- Fan：指风扇，一个场景中至少有一台风扇
- Hat：指帽子，一个场景中只能有一个帽子
- Actor：指角色，一个场景中只能有一个角色
- Barrier：指障碍，一个场景对障碍的个数不作要求

2.2 面向对象

游戏设计中所有数据对象，UI 对象都用类表示，类之间有继承，并且使用多态以进行组织。

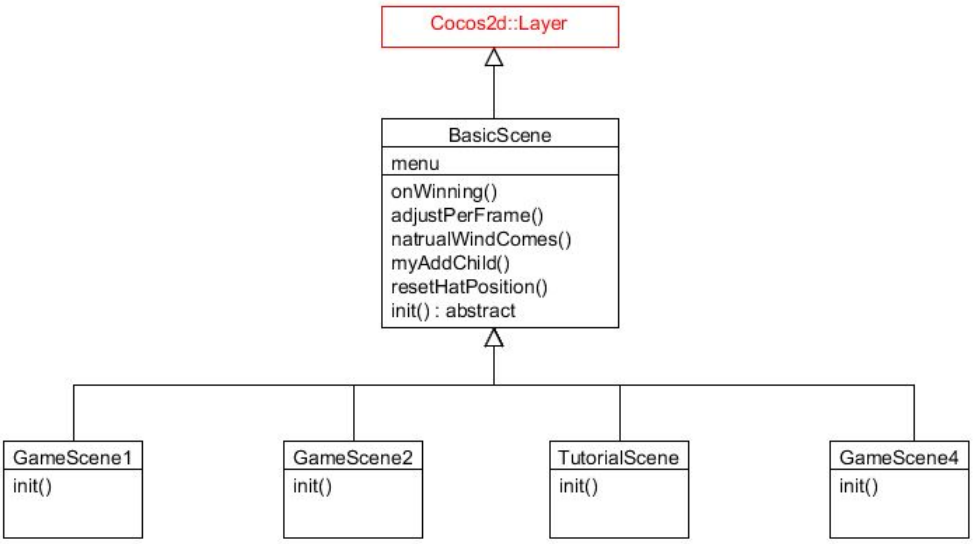
主要类分析

Game Scene 族

- Cocos2d::Layer 类：Cocos2d-x 引擎所提供的类，表示显示的一个界面。（红色表示不是由我们开发人员所编写的类）

- BasicScene 类：所有游戏场景的基类，它实现了所有游戏场景都需要的功能，例如重玩与返回菜单，胜利信息板，添加子节点等等。

- GameScene / TutorialScene 类：继承自 BasicScene，并会重写 init（）函数，让自己有自己的初始化内容。



代码表示：

```
class BasicScene : public Layer
{
public:
    virtual bool init();
    CREATE_FUNC(BasicScene);

public:
    // 胜利时调用的函数，播放胜利动画并返回选关场景
    void onWinning();

    // 每一帧的回调
    void update(float dt);
    void adjustPerFrame();

    // 每5秒的自然风
    void natrualWindComes(float dt);

    // 添加子节点
    void myAddChild(Sprite* s, const int type);

    // 返回按钮回调
    void onBackCallBack(Ref* pSender);
    void onReplayCallBack(Ref* pSender);

    virtual void resetHatPosition() {}

    ~BasicScene();
```

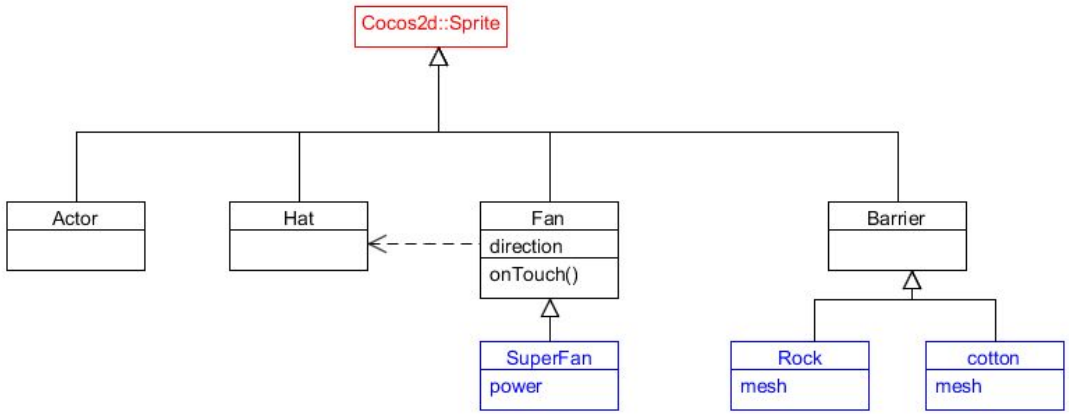
BasicScene.h

```
class GameScene1 : public BasicScene
{
private:
    Vector<Drama*> dramaList;
public:
    virtual bool init();
    virtual void resetHatPosition();
    static Scene* createScene();
    CREATE_FUNC(GameScene1);
};
```

GameScene1.h

Sprite 族

- Sprite 类：由 Cocos2d-x 引擎提供的精灵类，表示任何可以显示并做动作的物体（红色表示不是由我们开发人员所编写的类）
- Actor 类：继承自 Sprite，表示角色，他的功能是表示他的胜利区域，即当帽子与 Actor 的头部触碰时，就出发胜利事件。
- Hat 类：继承自 Sprite，指帽子，受风吹事件影响的帽子对象。
- Fan 类：风扇，具有方向，方向表示吹风朝向，它必须有点击事件响应函数。因为相应函数实现的是对帽子的吹动，因此 Fan 类对 Hat 类具有依赖关系
- Barrier 类：指障碍，当帽子和障碍碰撞时会产生一些物理效应，如反弹等。
- SuperFan 类：继承自 Fan 类，增多 power 属性，实现强力风扇功能（蓝色表示将来可能扩展的类）
- Rock 类：继承自 Barrier 类，重写 create 函数，可以改变其物理属性。（蓝色表示将来可能扩展的类）
- Cotton 类：继承自 Barrier 类，重写 create 函数，可以改变其物理属性。（蓝色表示将来可能扩展的类）



代码表示：

```
class Actor : public Sprite
{
public:
    Actor(void);
    ~Actor(void);

    static Actor* create(int actor_num);
};

#endif
```

Actor.h

```
class Hat : public Sprite
{
public:
    Hat(void);
    ~Hat(void);

    void setVelocity(Vec2 velocity);
    void onRemove();
    void removeItself();
    void comeOut();

    static Hat* create(int hat_num);

    int hat_num;
};
```

Hat.h

```
class Fan : public Sprite
{
public:
    Fan(void);
    ~Fan(void);

    /* create函数功能如下：
    根据输入的direction来创建具有相应图片的Fan实例。*/
    static Fan* create(const int direction);

    //开始触摸
    virtual bool onTouchBegan(Touch *pTouch, Event *pEvent);
    //触摸结束
    virtual void onTouchEnded(Touch *pTouch, Event *pEvent);


    // 指示方向
    static const int NORTH = 1;
    static const int NORTH_EAST = 2;
    static const int EAST = 3;
    static const int SOUTH_EAST = 4;
    static const int SOUTH = 5;
    static const int SOUTH_WEST = 6;
    static const int WEST = 7;
    static const int NORTH_WEST = 8;
    void setDirection(int d);

private:
    int direction;
};
```

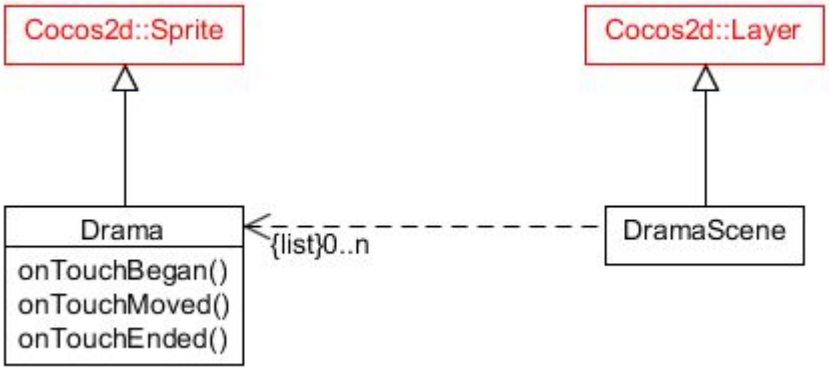
Fan.h

```
class Barrier : public Sprite
{
public:
    Barrier(void);
    ~Barrier(void);

    static Barrier* create(char* filename, int type);
    void setVelocity(int x, int y);
};
```

Barrier.h

Drama 族

- Sprite 类：如之前所述
- Layer 类：如之前所述
- Drama 类：继承自 Sprite 类，是剧情图片的载体，其本身必须实现拖动相应函数，让用户用拖动的方式来过剧情。
- DramaScene：继承自 Layer 类，是剧情的场景，也可以直接称之为剧情，它包含所有需要在此剧情中显示的 Drama 对象。



代码表示：

```
class Drama : public Sprite
{
public:
    static Drama* create(char* filename);

    virtual bool onTouchBegan(Touch *pTouch, Event *pEvent);
    virtual void onTouchMoved(Touch *pTouch, Event *pEvent);
    virtual void onTouchEnded(Touch *pTouch, Event *pEvent);

    void ActionEnded();

private:
    bool is_touch;
    Vec2 origin_pos;
};
```

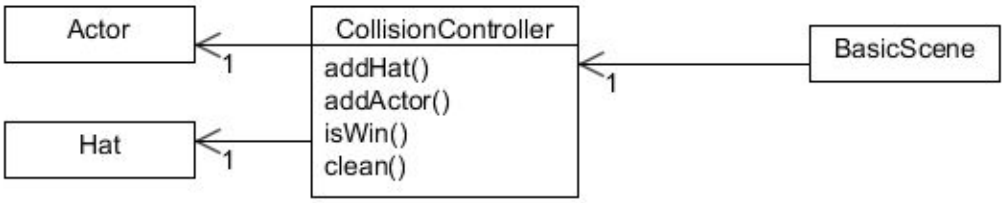
Drama.h

```
class DramaScene : public Layer
{
public:
    static Scene* createScene();
    virtual bool init();
    void update(float dt);
    CREATE_FUNC(DramaScene);
};
```

DramaScene.h

控制族：

- 控制族只有一个类，用于实现碰撞检测，实现一些碰撞的判断逻辑。可以看出，CollisionController 是用于检测 Actor 和 Hat 的碰撞的，他包含 Actor 和 Hat 实例。当然，它必须被 BasicScene 包含，因为在每一个 GameScene 中，都需要检测碰撞。

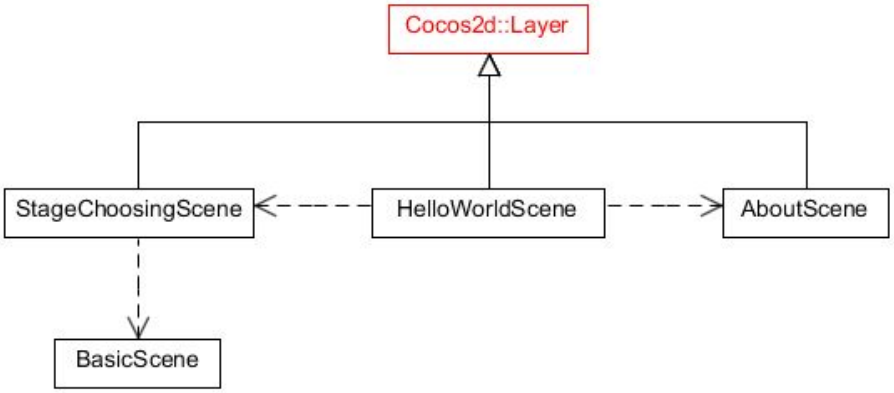


代码表示：

```
class CollisionController
{
public:
    CollisionController(void) : hat(nullptr), actor(nullptr) {}
    ~CollisionController(void);

    // 添加Hat实例指针
    void addHat(Hat* h);

    // 添加Actor实例指针
    void addActor(Actor* a);

    // 检查是否达到胜利条件
    bool isWin();

    void clean();


private:
    Hat* hat;
    Actor* actor;
};
```

普通 Scene 族：

- HelloworldScene 类：程序的第一个场景，供开始游戏之用，不带游戏逻辑，他可以引导用户到 StageChoosingScene 和 AboutScene 和 TutorialScene（属于 GameScene 族）。
- StageChoosingScene 类：选关场景，让用户选择关卡。
- AboutScene 类：关于场景，显示开发小组的信息。



代码表示：

```
class HelloWorld : public Layer
{
public:
    int selected_state;


    static cocos2d::Scene* createScene();


    virtual bool init();

    // a selector callback
    void menuCloseCallback(Ref* pSender);

    void onStartCallBack(Ref* pSender);

    void onTutorialCallBack(Ref* pSender);

    void onAboutCallBack(Ref* pSender);

    // implement the "static create()" method manually
    CREATE_FUNC(HelloWorld);
};
```

HelloWorldScene.h

```
class AboutScene : public Layer
{
public:
    static Scene* createScene();
    virtual bool init();

    void onBack(Ref* pSender);

    CREATE_FUNC(AboutScene);
};
```

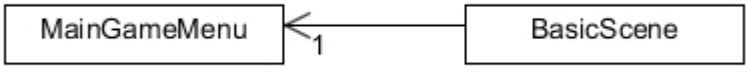
AboutScene.h

```
class StageChoosingScene : public Layer
{
public:
    static int stageNum;

public:
    static Scene* createScene();

    virtual bool init();


    void initMenu();


    void onLuffyStage1(Ref* pSender);
    void onLuffyStage2(Ref* pSender);
    void onWindComesStage1(Ref* pSender);
    void onBack(Ref* pSender);

    CREATE_FUNC(StageChoosingScene);
};
```

StageChoosingScene.h

MainGameMenu 类：

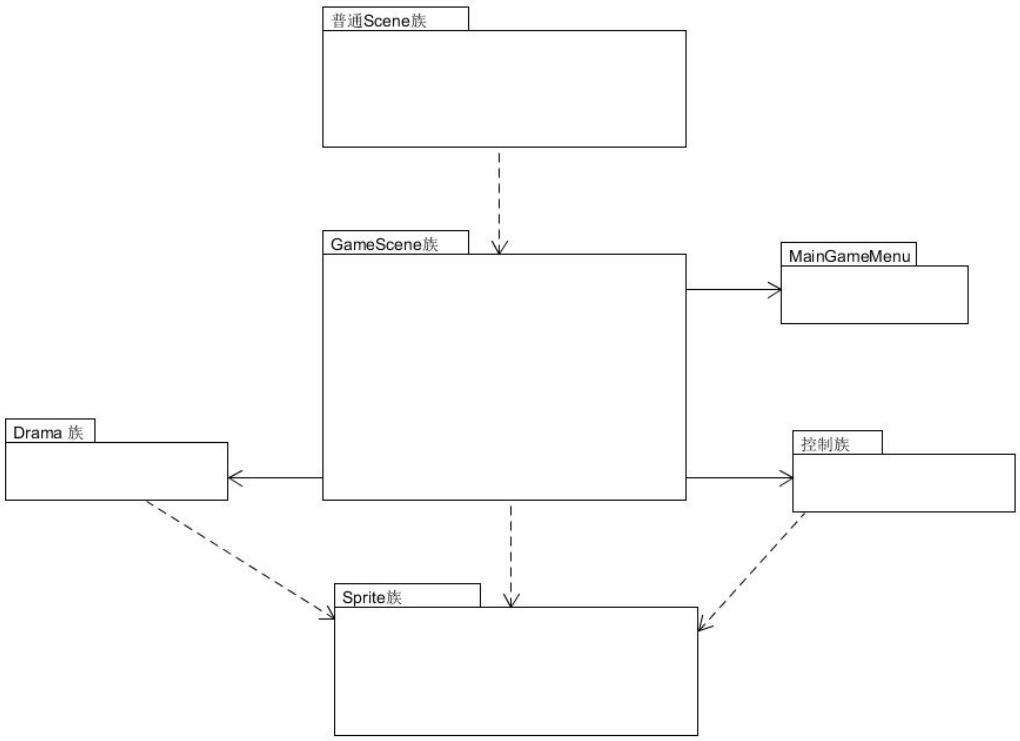
- MainGameMenu 类实现主游戏逻辑的两个按钮，重玩与返回按钮，当然，他被包含在 BasicScene 中。



代码表示：

```
class MainGameMenu
{
public:
    MainGameMenu(void) : game_menu(nullptr) {}
    ~MainGameMenu(void);

    // 初始化game_menu，向game_menu中添加MenuItem并且为MenuItem注册回调函数
    bool init();

    Menu* getGameMenu();

    void onBackItem(Ref* pSender);
    void onReplayItem(Ref* pSender);

private:
    Menu* game_menu;
};
```

2.3 模块合作



- 由普通 Scene 族做游戏的入口，普通 Scene 族依赖 GameScene 族进入主游戏场景，GameScene 族依赖 Sprite 族来构建场景，它还包含 MainGameMenu，Drama 族和控制族来进行一些必要的逻辑和进度工作。

3. Cocos2d-x 介绍

3.1 选用原因

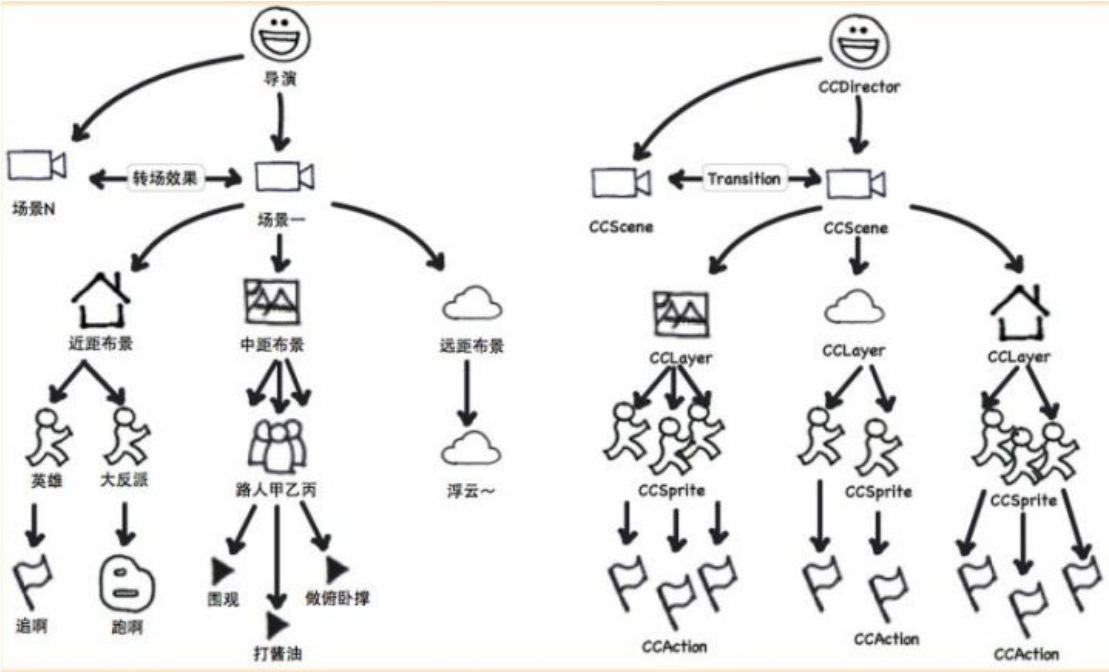
Cocos2d-x 的优势有很多，让我们选择他的理由如下：

- **开源，免费：**它很方便的为我们提供开源库，我们可以直接下载到电脑上，然后进行开饭。同时，开源的好处就是能够让我们更好的了解他，对于一些更高级的编程人员，还可以对他进行修改，以成为自己的一套开发工具。

- **跨平台：**跨平台是很重要的，虽然我们现在的主打平台是安卓，但我们不保证不会在别的平台上发布，而且跨平台的特性使其调试起来很简单，直接用 VS 就可以调试，而无须向 ADT 那样需要用虚拟设备或者真实的移动设备才能调试。

- **成熟：**市面上很多移动平台游戏都使用 Cocos2d-x 开发，因此很多技术在网上都能找到，很多问题也都能够在网上得到很好的解答，学习起来相对容易。

3.2 基本运作



由导演监管整个游戏，游戏在场景之间切换，场景有布景，布景上存放着许许多多的物体，即游戏元素。

4.测试

游戏的输入被 GUI 限制，即游戏只支持用户的几种输入，他们分别是：

- 点击
- 拖动

我们针对每个我们所能够看到的场景进行测试

1.**起始场景**，起始场景有三个按钮，Start，Tutorial，和 About。

测试用例	预期动作	实际动作
在背景处点击	无反应	无反应
在背景处拖动	无反应	无反应
点击 Start	进入选关场景	进入选关场景
点击 Tutorial	进入教学场景	进入教学场景
点击 About	进入关于场景	进入关于场景
在 Start 内拖动	进入选关场景	进入选关场景
在 Tutorial 内拖动	进入教学场景	进入教学场景
在 About 内拖动	进入关于场景	进入关于场景
从 Start 拖动到背景	无反应	无反应
从 Tutorial 拖动到背景	无反应	无反应
从 About 拖动到背景	无反应	无反应
从别的按钮拖动到 Start	进入选关场景	进入选关场景
从别的按钮拖动到 Tutorial	进入教学场景	进入教学场景
从别的按钮拖动到 About	进入关于场景	进入关于场景

2.选关场景：目前的选关场景有 3 个关卡按钮，分别为 stage1，stage2 和 stage3，还有一个 back 按钮。

测试用例	预期动作	实际动作
在背景处点击	无反应	无反应
在背景处拖动	无反应	无反应
点击 stage1	进入第一关场景	进入第一关场景
点击 stage2	进入第二关场景	进入第二关场景
点击 stage3	进入第三关场景	进入第三关场景
点击 Back	返回开始场景	返回开始场景
在 stage1 内拖动	进入第一关场景	进入第一关场景
在 stage2 内拖动	进入第二关场景	进入第二关场景
在 stage3 内拖动	进入第三关场景	进入第三关场景
从任意按钮拖动到背景	无反应	无反应
从背景拖动到任意按钮	无反应	无反应
从别的按钮拖动到 stage1	进入第一关场景	进入第一关场景
从别的按钮拖动到 stage2	进入第二关场景	进入第二关场景
从别的按钮拖动到 stage3	进入第三关场景	进入第三关场景

从别的按钮拖动到 Back	返回开始场景	返回开始场景

3.游戏内场景

游戏场景包括：帽子，风扇，人物，障碍，重玩按钮和返回按钮

测试用例	预期动作	实际动作
在背景处点击	无反应	无反应
在背景处拖动	无反应	无反应
点击帽子	无反应	无反应
拖动帽子到任意位置	无反应	无反应
从任意位置拖动到帽子	无反应	无反应
点击风扇	1.1 如果帽子在区域内，帽子朝风扇的方向被吹动。 1.2 如果帽子不在区域内，无反应	1.1 如果帽子在区域内，帽子朝风扇的方向被吹动。 1.2 如果帽子不在区域内，无反应
在风扇内拖动	1.1 如果帽子在区域内，帽子朝风扇的方向被吹动。 1.2 如果帽子不在区域内，无反应	1.1 如果帽子在区域内，帽子朝风扇的方向被吹动。 1.2 如果帽子不在区域内，无反应
点击人物	无反应	无反应
从任意位置拖动到人物	无反应	无反应
从人物拖动到任意位置	无反应	无反应
点击障碍	无反应	无反应
从任意位置拖动到障碍	无反应	无反应
从障碍拖动到任意位置	无反应	无反应
点击重玩按钮	场景所有物体都被重放会原来位置，执行原来的动作	场景所有物体都被重放会原来位置，执行原来的动作
点击返回按钮	返回选关场景	返回选关场景
从返回按钮拖动到重玩按钮	场景所有物体都被重放会原来位置，执行原来的动作	场景所有物体都被重放会原来位置，执行原来的动作
从重玩按钮拖动到返回按钮	返回选关场景	返回选关场景
程序 5 秒计数器到时	帽子被向上吹，表现为在垂直方向上有一个速度	帽子被向上吹，表现为在垂直方向上有一个速度
帽子与角色头部上半部分碰触	胜利，弹出胜利提示	胜利，弹出胜利提示
帽子与游戏场景边界碰撞	帽子反弹	帽子反弹
风扇碰撞	帽子反弹，风扇不动	帽子反弹，风扇不动
帽子与可移动障碍碰撞	双方反弹	双反反弹
帽子与不可移动障碍碰撞	帽子反弹	帽子反弹
点击胜利提示中的 ok	返回选关场景	返回选关场景

4.教学场景

游戏场景包括：帽子，风扇，人物，重玩按钮和返回按钮

测试用例	预期动作	实际动作
在背景处点击	无反应	无反应
在背景处拖动	无反应	无反应
点击帽子	无反应	无反应
拖动帽子到任意位置	无反应	无反应
从任意位置拖动到帽子	无反应	无反应
点击风扇	1.1 如果帽子在区域内，帽子朝风扇的方向被吹动。 1.2 如果帽子不在区域内，无反应	1.1 如果帽子在区域内，帽子朝风扇的方向被吹动。 1.2 如果帽子不在区域内，无反应
在风扇内拖动	1.1 如果帽子在区域内，帽子朝风扇的方向被吹动。 1.2 如果帽子不在区域内，无反应	1.1 如果帽子在区域内，帽子朝风扇的方向被吹动。 1.2 如果帽子不在区域内，无反应
点击人物	无反应	无反应
从任意位置拖动到人物	无反应	无反应
从人物拖动到任意位置	无反应	无反应
点击重玩按钮	场景所有物体都被重放会原来位置，执行原来的动作	场景所有物体都被重放会原来位置，执行原来的动作
点击返回按钮	返回开始场景	返回开始场景
从返回按钮拖动到重玩按钮	场景所有物体都被重放会原来位置，执行原来的动作	场景所有物体都被重放会原来位置，执行原来的动作
从重玩按钮拖动到返回按钮	返回开始场景	返回开始场景
程序 5 秒计数器到时	帽子被向上吹，表现为在垂直方向上有一个速度	帽子被向上吹，表现为在垂直方向上有一个速度
帽子与角色头部上半部分碰触	胜利，弹出胜利提示	胜利，弹出胜利提示
帽子与游戏场景边界碰撞	帽子反弹	帽子反弹
风扇碰撞	帽子反弹，风扇不动	帽子反弹，风扇不动
帽子与可移动障碍碰撞	双方反弹	双反反弹
帽子与不可移动障碍碰撞	帽子反弹	帽子反弹
点击胜利提示中的 ok	返回选关场景	返回选关场景

5.关于场景

关于场景包括：背景和一个 Back 返回按钮

测试用例	预期动作	实际动作
在背景处点击	无反应	无反应
在背景处拖动	无反应	无反应
点击返回按钮	返回开始场景	返回开始场景

5.结束

《等风来》是一款由等风来小组独立开发的安卓端手机游戏。

开发人员：潘文优，马东宇，李喆，梁嘉升，林炽锦，彭够妹