程序清单

--基于微信开发者工具开发的飞翔的小鸟游戏



课 程： 软件工程导论

名 称： Flappy Bird

人 员: G17小组

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订日期 | 版本/状态 | 修订人 | 备注 |
| 1 | 2020/1/3 | 1.0 | 李以昕 | 第一次正式版 |

目录

[1.全局函数 5](#_Toc60574913)

[1.1全局变量 5](#_Toc60574914)

[1.2请求调用云函数 6](#_Toc60574915)

[1.3图片加载 12](#_Toc60574916)

[1.4 界面跳转 13](#_Toc60574917)

[1.5 更换角色图片 13](#_Toc60574918)

[2.游戏模块 14](#_Toc60574919)

[2.1 登录游戏 14](#_Toc60574920)

[2.2开始游戏 15](#_Toc60574921)

[2.2.1难度选择 16](#_Toc60574922)

[2.2.2 地图选择 17](#_Toc60574923)

[2.2.3 游戏加载 17](#_Toc60574924)

[2.3游戏内容 18](#_Toc60574925)

[2.3.1小鸟动作 19](#_Toc60574926)

[2.3.2游戏进程 20](#_Toc60574927)

[2.3.3金币逻辑 23](#_Toc60574928)

[2.3.4 金币显示 25](#_Toc60574929)

[2.3.5 水管显示 25](#_Toc60574930)

[2.4 游戏结束 27](#_Toc60574931)

[3好友与收件箱 28](#_Toc60574932)

[3.1 Show Friends 28](#_Toc60574933)

[3.1Loading Friends 30](#_Toc60574934)

[4.商店与角色模块 33](#_Toc60574935)

[4.1商店 33](#_Toc60574936)

[4.1.1 加载角色预览 33](#_Toc60574937)

[4.1.2 角色信息 35](#_Toc60574938)

[4.1.3 角色购买 37](#_Toc60574939)

[4.2 角色 39](#_Toc60574940)

[4.2.1 加载角色预览 39](#_Toc60574941)

[4.2.2 拥有角色信息 42](#_Toc60574942)

[4.2.3 更换角色 44](#_Toc60574943)

[5.排行榜模块 45](#_Toc60574944)

[5.1 加载世界排行榜 45](#_Toc60574945)

[5.2加载玩家成绩信息 47](#_Toc60574946)

[5.3 取消举报 48](#_Toc60574947)

[5.4确定举报 49](#_Toc60574948)

[5.5 举报界面 50](#_Toc60574949)

[6局部变量 51](#_Toc60574950)

[6.1游戏难度系数 51](#_Toc60574951)

[6.2游戏界面 51](#_Toc60574952)

[6.3 跳转难度界面 52](#_Toc60574953)

[6.4 排行榜界面 53](#_Toc60574954)

[6.5举报界面 53](#_Toc60574955)

[6.6商店角色界面 53](#_Toc60574956)

[6.7用户拥有角色界面 54](#_Toc60574957)

[7.云函数 54](#_Toc60574958)

[7.1角色 54](#_Toc60574959)

[7.2游戏 55](#_Toc60574960)

[7.3加载小鸟图片 57](#_Toc60574961)

[7.4加载游戏 58](#_Toc60574962)

[7.5加载世界排行榜 59](#_Toc60574963)

[7.6商店 60](#_Toc60574964)

[7.7商店购买 61](#_Toc60574965)

# 1.全局函数

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 全局变量 | Global\_Variable |
| 2 | 请求调用云函数 | Global\_WeChat |
| 3 | 图片加载 | Cloud\_Image |
| 4 | 场景跳转 | Jump\_Scene |
| 5 | 加载资源 | Loading\_Base\_Resouce |
| 6 | 更换角色图片 | Change\_Bird\_Image |

## 1.1全局变量

window.Global\_Variable = {

    User\_Information: null,

    openid: null,

    Gold: 0,

    Diamond: 0,

    Compassion: 0,

    User\_Head\_Image: null,

    User\_Name: null,

    Best\_Score: 0,

    Restores\_Compassion\_Time: new Date(),

    Character\_Image1: null,

};

## 1.2请求调用云函数

//全局变量

var Rank\_Local\_Varible = require('Rank\_Local\_Variable');

var Shop\_Character\_Local\_Variable=require('Shop\_Character\_Local\_Varible');

var User\_Have\_Character\_Local\_Varible=require('User\_Have\_Character\_Local\_Varible');

window.WeChat = {};

//微信登录注册调用

WeChat.onRegisterUser = function(\_userinfo) {

    wx.cloud.init({

        env: "zcx-6gbgdxdy254816b0"

    })

    console.log(\_userinfo);

    wx.cloud.callFunction({

        //调用云函数

        name: "login",

        //传入的参数，玩家信息

        data: {

            userinfo: \_userinfo,

        },

        success(res) {

            cc.director.loadScene("Game\_Start");

            console.log("登录注册成功回调", res);

        },

        fail: console.log("程序出错", console.error)

    })

}

//加载资源

WeChat.Loading\_Resources = function() {

    wx.cloud.init({

        env: "zcx-6gbgdxdy254816b0"

    })

    wx.cloud.callFunction({

        //调用云函数

        name: "Loading\_Resources",

        success(res) {

            //将值赋给全局变量

            console.log("获取成功回调", res);

            Global\_Variable.openid = res.result.openid;

            Global\_Variable.Gold = res.result.Gold;

            Global\_Variable.Diamond = res.result.Diamond;

            Global\_Variable.Compassion = res.result.Compassion;

            Global\_Variable.User\_Head\_Image = res.result.User\_Head\_Image;

            Global\_Variable.Best\_Score = res.result.Best\_Score;

            Global\_Variable.User\_Name = res.result.User\_Name;

            Global\_Variable.Current\_Character\_Id=res.result.Current\_Character\_Id;

            WeChat.Loading\_Bird\_Image(Global\_Variable.Current\_Character\_Id);

        },

        fail() {

            console.log("获取基础物资出错", console.error);

            //失败了重新赋值

            WeChat.Loading\_Resources();

        }

    })

}

//游戏结算

WeChat.Game\_Settlement = function(\_Add\_Gold, \_Score) {

    wx.cloud.init({

        env: "zcx-6gbgdxdy254816b0"

    })

    wx.cloud.callFunction({

        //调用云函数

        name: "Game\_Settlement",

        //传入参数，包括增加的金币数量，分数

        data: {

            Add\_Gold: \_Add\_Gold,

            Score: \_Score,

        },

        success(res) {

            console.log("获取成功回调", res);

            Global\_Variable.Gold = res.result.Gold;

            Global\_Variable.Diamond = res.result.Diamond;

            Global\_Variable.Compassion = res.result.Compassion;

        },

        fail() {

            console.log("获取基础物资出错", console.error);

            WeChat.Loading\_Resources();

        }

    })

}

//加载世界排名

WeChat.Loading\_World\_Rank = function() {

    wx.cloud.init({

        env: "zcx-6gbgdxdy254816b0"

    })

    wx.cloud.callFunction({

        //调用云函数

        name: "Loading\_World\_Rank",

        success(res) {

            console.log("下载世界排名成功回调", res);

            Rank\_Local\_Varible.Word\_Rank\_User = res.result.User\_Information.data;

            console.log("Rank\_Local\_Varible为", Rank\_Local\_Varible.Word\_Rank\_User[0].openid);

        },

        fail() {

            console.log("下载世界排名出错", console.error);

            WeChat.Loading\_Resources();

        }

    })

}

//上传举报列表

WeChat.Uploading\_Reported\_Information = function(\_Reported\_Openid, \_Report\_Content) {

    wx.cloud.init({

        env: "zcx-6gbgdxdy254816b0"

    })

    wx.cloud.callFunction({

        //调用云函数

        name: "Uploading\_Reported\_Information",

        data: {

            Reported\_Openid: \_Reported\_Openid,

            Report\_Content: \_Report\_Content,

        },

        success(res) {

            console.log("举报成功回调", res);

        },

        fail() {

            console.log("举报出错回调", console.error);

        }

    })

}

WeChat.Loading\_Character=function () {

    wx.cloud.init({env:"zcx-6gbgdxdy254816b0"})

    wx.cloud.callFunction({

        //调用云函数

        //传入的参数

        name:"Character",

        success(res){

            console.log("获取角色信息成功",res);

            Shop\_Character\_Local\_Variable.Shop\_Character\_User=res.result.Character\_Information.data;

            User\_Have\_Character\_Local\_Varible.User\_Have\_Character=res.result.User\_Have\_Character\_Information.data;

        },

        fail(){

            console.log("获取角色信息失败",console.error);

            WeChat.Loading\_Resources();

        }

    })

}

WeChat.Loading\_Shop\_Character=function(){

    wx.cloud.init({env:"zcx-6gbgdxdy254816b0"})

    wx.cloud.callFunction({

        //调用云函数

        //传入的参数

        name:"Shop",

        success(res){

            console.log("获取商店信息成功",res);

            Shop\_Character\_Local\_Variable.Shop\_Character\_User=res.result.Character\_Information.data;

            User\_Have\_Character\_Local\_Varible.User\_Have\_Character=res.result.User\_Have\_Character\_Information.data;

        },

        fail(){

            console.log("获取商店信息失败",console.error);

            WeChat.Loading\_Resources();

        }

    })

}

//商店购买跟新

WeChat.Buy\_Character\_Update=function (\_Update\_Gold,\_Update\_Character) {

    wx.cloud.init({env:"zcx-6gbgdxdy254816b0"})

    wx.cloud.callFunction({

        //调用云函数

        //传入的参数

        name:"Shop\_Buy",

        data:{

            Update\_Gold:\_Update\_Gold,

            Update\_Character:\_Update\_Character,

        },

        success(res){

            console.log("商店获取成功回调",res);

        },

        fail(){

            console.log("商店获回调出错",console.error);

            WeChat.Loading\_Resources();

        }

    })

}

WeChat.Loading\_Bird\_Image=function (Character\_Id) {

    wx.cloud.init({env:"zcx-6gbgdxdy254816b0"})

    wx.cloud.callFunction({

        //调用云函数

        //传入的参数

        name:"Loading\_Bird\_Image",

        data:{

            Character\_Id:Character\_Id,

        },

        success(res){

            console.log("角色图像获取成功回调",res);

            cc.loader.load({

                url:res.result.Character\_Image1,

                type:'jpg'

            },function(err,texture,test){

                var frame=new cc.SpriteFrame(texture);

                if(err){

                    console.log("图片错误",err);

                }

                Global\_Variable.Character\_Image1=frame;

            })

        },

        fail(){

            console.log("角色图像回调出错",console.error);

        }

    })

}

WeChat.Updating\_Current\_Character\_id=function (Character\_Id) {

    wx.cloud.init({env:"zcx-6gbgdxdy254816b0"})

    wx.cloud.callFunction({

        //调用云函数

        //传入的参数

        name:"Updating\_Current\_Character\_id",

        data:{

            Character\_Id:Character\_Id,

        },

        success(res){

            console.log("修改当前角色成功回调",res);

            WeChat.Loading\_Bird\_Image(Character\_Id);

        },

        fail(){

            console.log("修改当前角色失败",console.error);

            WeChat.Loading\_Bird\_Image(Character\_Id);

        }

    })

}

## 1.3图片加载

//改变图片

cc.Class({

    extends: cc.Component,

    properties: {

        Iamge: "",//图片所在路径

        BGSprite: {

            default: null,

            type: cc.Sprite,

            serialzable: true,

        }, //图片要加载的组件

    },

    // onLoad () {},

    onLoad: function() {

        var self = this;

        let \_url = self.Iamge;

        //下载图片

        cc.loader.load({

            url: \_url,

            type: 'jpg'

        }, function(err, texture, test) {

            var frame = new cc.SpriteFrame(texture);

            if (err) {

                console.log("图片错误", err);

            }

            self.BGSprite.getComponent(cc.Sprite).spriteFrame = frame;

        })

    },

    update: function(dt) {},

});

## 1.4 界面跳转

//界面跳转

cc.Class({

    extends: cc.Component,

    properties: {

        scene: "",

    },

    start() {

    },

    on\_btn\_click: function() {

        cc.director.loadScene(this.scene);

    }

});

## 1.5 更换角色图片

cc.Class({

    extends: cc.Component,

    properties: {

    },

    start () {

    },

    update (dt) {

    if(Global\_Variable.Character\_Image1!=null)

    this.node.getComponent(cc.Sprite).spriteFrame = Global\_Variable.Character\_Image1;

  },

});

# 2.游戏模块

## 2.1 登录游戏

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 微信登录 | WX\_Login |

cc.Class({

    extends: cc.Component,

    properties: {

    },

    onLoad: function() {

        //调用微信登录函数，登录游戏

        wx.login({

            success: function(res) {

                if (res.code) {

                    console.log("登录成功，获取到code", res.code);

                }

                var button = wx.createUserInfoButton({

                    type: 'text',

                    text: '开始游戏',

                    style: {

                        left: wx.getSystemInfoSync().screenWidth / 2 - 60,

                        top: wx.getSystemInfoSync().screenHeight / 2 - 60,

                        width: 120,

                        height: 40,

                        lineHeight: 40,

                        backgroundColor: '#00aa00',

                        color: '#ffffff',

                        textAlign: 'center',

                        fontSize: 16,

                        borderRadius: 90,

                    }

                });

                button.hide();

                button.show();

                button.onTap((res) => {

                    console.log(res);

                    if (res.errMsg === "getUserInfo:ok") {

                        console.log("已经授权");

                        //获取注册信息

                        console.log(res.userInfo);

                        WeChat.onRegisterUser(res.userInfo);

                        button.destroy();

                        //cc.director.loadScene("Game\_Start");

                    } else {

                        console.log("没有授权");

                    }

                });

            }

        });

    },

    start() {

    },

    // update (dt) {},

});

## 2.2开始游戏

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 难度选择 | Change\_Difficult |
| 2 | 地图选择 | Change\_Map |
| 3 | 游戏加载 | Loading\_Game |

### 2.2.1难度选择

var Game\_Difficulty\_Local\_Varible = require('Game\_Difficulty\_Local\_Varible');

//改变游戏难度

cc.Class({

    extends: cc.Component,

    properties: {

        Difficulty: "", //声明游戏难度

    },

    start() {

    },

    // update (dt) {},

    on\_btn\_click: function() { //根据不同的游戏难度，调整难度系数

        if (this.Difficulty == 'Easy') {

            Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio = 0.5;

            Game\_Difficulty\_Local\_Varible.Is\_Difficulty = false;

            Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio = 1;

        } else if (this.Difficulty == 'Mid') {

            Game\_Difficulty\_Local\_Varible.Is\_Difficulty = false;

        } else if (this.Difficulty == 'Difficulty') {

            Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio = 1;

            Game\_Difficulty\_Local\_Varible.Is\_Difficulty = true;

        }

    }

});

### 2.2.2 地图选择

var Game\_Difficulty\_Local\_Varible = require('Game\_Difficulty\_Local\_Varible');

//改变游戏难度

cc.Class({

    extends: cc.Component,

    properties: {

        Difficulty: "", //声明游戏难度

    },

    start() {

    },

    // update (dt) {},

    on\_btn\_click: function() { //根据不同的游戏难度，调整难度系数

        if (this.Difficulty == 'Easy') {

            Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio = 0.5;

            Game\_Difficulty\_Local\_Varible.Is\_Difficulty = false;

            Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio = 1;

        } else if (this.Difficulty == 'Mid') {

            Game\_Difficulty\_Local\_Varible.Is\_Difficulty = false;

        } else if (this.Difficulty == 'Difficulty') {

            Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio = 1;

            Game\_Difficulty\_Local\_Varible.Is\_Difficulty = true;

        }

    }

});

### 2.2.3 游戏加载

//加载游戏资源

cc.Class({

    extends: cc.Component,

    properties: {

        Diamond\_Show: {

            default: null,

            type: cc.Label,

            serialzable: true,

        },

        Gold\_Show: {

            default: null,

            type: cc.Label,

            serialzable: true,

        },

        Compassion\_Show: {

            default: null,

            type: cc.Label,

            serialzable: true,

        }

    },

    onLoad: function() {

        WeChat.Loading\_Resources();

    },

    start: function() {},

    update: function(dt) {

        //改变导航栏的基础资源数

        this.Gold\_Show.string = Global\_Variable.Gold;

        this.Diamond\_Show.string = Global\_Variable.Diamond;

        this.Compassion\_Show.string = Global\_Variable.Compassion;

    },

});

## 2.3游戏内容

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 小鸟动作 | Bird\_Action |
| 2 | 游戏进程 | Game |
| 3 | 金币逻辑 | Gold\_Action |
| 4 | 金币数据显示 | Gold\_Show\_Action |
| 5 | 水管显示 | Waterpipe\_Action |

### 2.3.1小鸟动作

//控制小鸟的运动情况

var Game\_Difficulty\_Local\_Varible = require('Game\_Difficulty\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        \_V\_Index: 0, //小鸟垂直速度

        \_Is\_Sart: true, //游戏是否进行

        Jump\_Up\_Acc: 0, //上升速度

        Jump\_Down\_Acc: 0,

    },

    onLoad: function() {

        //开启小鸟的碰撞属性

        var manager = cc.director.getCollisionManager();

        //manager.enabledDebugDraw = true;

        manager.enabled = true;

        //监听屏幕是否被点击

        this.node.parent.on(cc.Node.EventType.TOUCH\_START, this.onTouchMove, this);

    },

    onDestroy() {

        // 取消键盘输入监听

    },

    start() {

    },

    update: function(dt) {

        //如果游戏开始，是的小鸟开始掉落

        if (this.\_Is\_Sart) {

            //随着时间的增长，增加下降速度。

            this.\_V\_Index -= this.Jump\_Down\_Acc \* dt;

        }

        //通过改变小鸟坐标的形式，使得小鸟移动

        this.node.y += this.\_V\_Index \* dt;

    },

    onTouchMove(event) {

        //如果屏幕被点击，给予小鸟一个向上的加速度

        this.\_V\_Index = this.Jump\_Up\_Acc \* 20;

    },

    onCollisionEnter: function(other, self) {

        //如果发生碰撞，调用函数

        console.log("other.name = ", other.node.name, other.node.group, other.node.groupIndex);

        if (other.node.groupIndex === 1) { // 与障碍物相撞

            this.\_Is\_Sart = false;

            this.\_V\_Index = 3000;

            //使得小鸟向上飞出屏幕

        }

    },

});

### 2.3.2游戏进程

//控制游戏进程的JS脚本

var Game\_Local\_Varible = require('Game\_Local\_Varible');

var Game\_Difficulty\_Local\_Varible = require('Game\_Difficulty\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Bird: {

            default: null,

            type: cc.Sprite,

            serialzable: true,

        }, //载入小鸟

        Background: {

            default: null,

            type: cc.Sprite,

            serialzable: true,

        }, //载入背景

        Fraction: {

            default: null,

            type: cc.Label,

            serialzable: true,

        }, //载入分数统计

        Waterpipe\_Up: {

            default: null,

            type: cc.Prefab,

            serialzable: true,

        }, //载入上方水管的预制体

        Waterpipe\_Down: {

            default: null,

            type: cc.Prefab,

            serialzable: true,

        }, //载入下方水管的预制体

        Fall\_Gold: {

            default: null,

            type: cc.Prefab,

            serialzable: true,

        }, //载入金币的预制体

        Is\_Start: false, //控制游戏是否进行

        time: 0, //载入时间

        Time\_Label: {

            default: null,

            type: cc.Label,

            serialzable: true,

        }, //载入倒计的Label

        daojishi: 3, //设置倒计时时间

    },

    // LIFE-CYCLE CALLBACKS:

    // onLoad () {},

    onLoad: function() {

        Game\_Local\_Varible.Gold = 0;

        Game\_Local\_Varible.Fraction = 0;

        this.Bird.node.active = false;

        this.Background.getComponent(cc.Sprite).spriteFrame = Game\_Local\_Varible.Current\_Map;

    },

    start: function() {

        this.Time\_Label.string = 3; //  场景文本框为 显示3

        this.daojishi = 3;

        if (this.daojishi >= 0) {

            this.schedule(function() { // 计时器将每隔 1s 执行一次。

                this.DoSomething();

                if (this.daojishi == 0) {

                    this.Time\_Label.enabled = false;

                    this.Is\_Start = true;

                    this.Bird.node.active = true;

                }

            }, 1);

        }

    },

    update: function(dt) {

        this.time += 1;

        this.gainScore();

        //每隔一段时间生成一个水管

        if (this.time % 100 == 0 && this.Is\_Start) {

            this.spawnNewWaterpipe();

        }

        //如果小鸟飞出了界面，游戏结束，进行跳转。

        if (this.Bird.node.y > 2000) {

            cc.director.loadScene("Game\_Over");

        }

    },

    //随机生成新的水管

    spawnNewWaterpipe() {

        var Waterpipe\_Up\_randY = Math.random() \* 400 + 800 + Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio \* 200; //随机生成上方水管的坐标

        var Waterpipe\_Down\_randY = Waterpipe\_Up\_randY - Math.random() \* 200 - 2000 + 200 \* Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio; //随机生成下方水管的坐标

        var Gold\_randY = (Waterpipe\_Up\_randY + Waterpipe\_Down\_randY) / 2; //再水管中间生成金币的坐标

        var randX = 700; //生成水管的X坐标

        //生成新的水管和进行，并且把他们加入到画布中

        var New\_Waterpipe\_Up = cc.instantiate(this.Waterpipe\_Up);

        this.node.addChild(New\_Waterpipe\_Up);

        New\_Waterpipe\_Up.setPosition(randX, Waterpipe\_Up\_randY);

        var New\_Waterpipe\_Down = cc.instantiate(this.Waterpipe\_Down);

        this.node.addChild(New\_Waterpipe\_Down);

        New\_Waterpipe\_Down.setPosition(randX, Waterpipe\_Down\_randY);

        var New\_Fall\_Gold = cc.instantiate(this.Fall\_Gold);

        this.node.addChild(New\_Fall\_Gold);

        New\_Fall\_Gold.setPosition(randX, Gold\_randY);

    },

    gainScore() {

        // 更新 scoreDisplay Label 的文字

        this.Fraction.string = 'Score: ' + (Game\_Local\_Varible.Fraction / 2);

    },

    DoSomething() { // 倒计时算法

        if (this.daojishi >= 1) {

            this.daojishi = this.daojishi - 1;

            this.Time\_Label.string = this.daojishi; //场景中文本框显示

            //cc.log("daojishi=" + this.daojishi);

        }

    }

});

### 2.3.3金币逻辑

//金币运动函数

var Game\_Local\_Varible = require('Game\_Local\_Varible');

var Game\_Difficulty\_Local\_Varible = require('Game\_Difficulty\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Move\_Speed: 0, //金币的移动速度

        x: 0, //金币x坐标

        y: 0, //金币y坐标

        Gold\_Show: { //金币的预制体

            default: null,

            type: cc.Prefab,

            serialzable: true,

        },

    },

    // LIFE-CYCLE CALLBACKS:

    onLoad: function() {

        //打开碰撞属性

        var manager = cc.director.getCollisionManager();

        manager.enabled = true;

        //manager.enabledDebugDraw = true;

    },

    start: function() {},

    update: function(dt) {

        //通过更新金币的x坐标，让金币前进

        this.node.x -= (5 + Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio \* 2);

        //如果金币超出界面，毁灭掉金币

        if (this.node.x < -600) {

            this.node.destroy();

        }

    },

    //碰撞判断

    onCollisionEnter: function(other, self) {

        console.log("other.name = ", other.node.name, other.node.group, other.node.groupIndex);

        if (other.node.groupIndex === 0) { //如果和小鸟碰撞

            var Gold\_Show\_Label = cc.instantiate(this.Gold\_Show); //加入金币显示框

            Game\_Local\_Varible.Gold += 1;

            this.node.destroy(); //移除该金币

            this.node.parent.addChild(Gold\_Show\_Label);

            Gold\_Show\_Label.setPosition(400, 750);

        }

    },

});

### 2.3.4 金币显示

//金币显示框运动逻辑

var Game\_Local\_Varible = require('Game\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Gold\_Show:{  //金币显示框

            default:null,

            type:cc.Label,

            serialzable:true,

        },

    },

    // LIFE-CYCLE CALLBACKS:

    onLoad: function() {

        console.log("Game\_Local\_Varible.Gold="+Game\_Local\_Varible.Gold)

        this.Gold\_Show.string = (Game\_Local\_Varible.Gold);  //改变金币显示数

    },

    start () {

    },

    update: function(dt) {

        this.Gold\_Show.string = (Game\_Local\_Varible.Gold);//改变金币显示数

    },

});

### 2.3.5 水管显示

//水管移动逻辑

var Game\_Difficulty\_Local\_Varible = require('Game\_Difficulty\_Local\_Varible');

var Game\_Local\_Varible = require('Game\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Move\_Speed: 0, //水平移动速度

        x: 0, //x坐标

        y: 0, //y坐标

        Move\_Distance: 0, //移动的距离

        Is\_Move: false, //是否垂直移动

        Is\_Up: true, //是否向上移动 ，如果为否，就会向下移动

    },

    // LIFE-CYCLE CALLBACKS:

    onLoad: function() {

        var manager = cc.director.getCollisionManager(); //改变碰撞体积

        manager.enabled = true;

        //manager.enabledDebugDraw = true;

        this.Is\_Move = Game\_Difficulty\_Local\_Varible.Is\_Difficulty; //如果游戏难度为最难，则则不能上下移动

        if (Math.random() > 0.5) {

            this.Is\_Up = false; //决定上下移动还是左右移动

        }

    },

    start: function() {},

    update: function(dt) {

        this.node.x -= (5 + Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio \* 2); //水平移动水管

        if (this.node.x < -600) { //如果水管已出屏幕，分数加一，毁灭掉水管

            Game\_Local\_Varible.Fraction += 1;

            this.node.destroy();

        }

        if (this.Is\_Move == true && this.node.x < 0) { //如果水管能够上下移动，添加水管的上下移动距离

            this.Move\_Distance = 50;

            this.Is\_Move = false;

        }

        if (this.Move\_Distance > 0 && this.Is\_Up == true) { //使水管上下移动

            this.node.y += 5;

            this.Move\_Distance -= 5;

        } else if (this.Move\_Distance > 0 && this.Is\_Up == false) {

            this.node.y -= 5;

            this.Move\_Distance -= 5;

        }

    },

});

## 2.4 游戏结束

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 结算成绩 | Global\_Variable |

//进行游戏结束后的结算

var Game\_Local\_Varible = require('Game\_Local\_Varible');

var Game\_Difficulty\_Local\_Varible = require('Game\_Difficulty\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Grade\_Show: {

            default: null,

            type: cc.Label,

            serialzable: true,

        }, //调入金币显示的Label

        Fraction\_Show: {

            default: null,

            type: cc.Label,

            serialzable: true,

        }, //调入分数显示的Label

        Background: {

            default: null,

            type: cc.Sprite,

            serialzable: true,

        }, //调入分数显示的Lab

    },

    // LIFE-CYCLE CALLBACKS:

    onLoad: function() {

        WeChat.Game\_Settlement(Math.floor(Game\_Local\_Varible.Fraction / 4 \* Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio +

            Game\_Local\_Varible.Gold), Game\_Local\_Varible.Fraction / 2); //将分数传入云数据库中

        this.Grade\_Show.string = "一共通过了" + (Game\_Local\_Varible.Fraction / 2) + "根水管\n收获了" + (Math.floor(Game\_Local\_Varible

            .Fraction / 4 \* Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio + Game\_Local\_Varible.Gold)) + "枚金币"; //将Label更新

        this.Fraction\_Show.string = (Game\_Local\_Varible.Fraction / 2) \* Game\_Difficulty\_Local\_Varible.Difficulty\_Ratio; //更新前端Label

        this.Background.getComponent(cc.Sprite).spriteFrame = Game\_Local\_Varible.Current\_Map;

    },

    start() {},

    update: function(dt) {},

});

# 3好友与收件箱

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 好友列表 | Show\_Friend、Loading\_Friends |

## 3.1 Show Friends

//加载好友

cc.Class({

    extends: cc.Component,

    properties: {

        WXSubContextView: cc.SubContextView

    },

    onLoad: function() {

        console.log("显示排行榜");

        //获取时间戳

        let updateTime = parseInt(new Date().getTime() / 1000);

        let getArr = new Array(); //声明向开放数据域传递的变量

        let openDataContext = wx.getOpenDataContext(); //声明开发数据域

        let setArr = [{

            key: "score",

            value: String(Global\_Variable.Best\_Score)

        }];

        getArr.push("score"); //将score压入开放数据域

        //向开发数据域声明获取好友

        openDataContext.postMessage({

            type: "GET",

            data: getArr,

            timer: updateTime

        })

        //将自己的成绩送入开放数据域

        console.log(setArr);

        openDataContext.postMessage({

            type: "SET",

            data: setArr,

            timer: updateTime

        })

        console.log("设置的信息为", setArr);

        //显示开放数据域

        this.SubContextView = this.WXSubContextView.getComponent(cc.SubContextView);

        this.SubContextView.enabled = true;

        this.SubContextView.active = true;

        this.SubContextView.update();

        console.log("特别正常");

    },

    start() {

    },

    // update (dt) {},

});

## 3.1Loading Friends

//下载好友信息

cc.Class({

    extends: cc.Component,

    properties: {

        Friend\_Show\_Label: cc.Node,

        Friend\_Show: cc.Prefab,

    },

    // onLoad () {},

    start() {

        console.log("该子域被调用显示了");

        //接收主域信息

        wx.onMessage(reDate => {

            console.log(reDate);

            //如果为get，更新关系域中存储的信息。

            if (reDate.type == "GET") {

                console.log("good");

                let children = this.Friend\_Show\_Label.children;

                let length = children.length;

                //清空列表

                for (let i = 0; i < length; i++) {

                    children[i].destroy();

                }

                console.log("good");

                wx.getFriendCloudStorage({

                    keyList: reDate.data,

                    success: res => {

                        console.log("好友信息：", res.data, reDate.data);

                        let \_tempArr = [];

                        //将好友信息载入\_tempArr中

                        for (let i = 0; i < res.data.length; ++i) {

                            console.log("score为", res.data[i].KVDataList[0].value);

                            let \_parseScore = Number(res.data[i].KVDataList[0].value);

                            \_tempArr.push(\_parseScore);

                        }

                        //排序

                        \_tempArr.sort((a, b) => {

                            return b - a;

                        });

                        //匹配

                        let index = 0;

                        while (index < \_tempArr.length) {

                            //根据排序后的数值生成排行榜数据

                            for (let i = 0; i < res.data.length; ++i) {

                                let \_parseScore = Number(res.data[i].KVDataList[0].value);

                                if (\_tempArr[index] == \_parseScore) {

                                    this.Create\_User\_Block(res.data[i], index);

                                    res.data.splice(i, 1);

                                    break;

                                }

                            }

                            index++;

                        }

                    }

                })

            } else if (reDate.type == "SET") {

                //接收信息，更新子域中存储的信息

                wx.setUserCloudStorage({

                    KVDataList: reDate.data,

                    success: function(res) {

                        console.log("存储成功");

                        console.log(res, reDate.data);

                    },

                    fail: function(res) {

                        console.log("存储失败");

                        console.error(res);

                    },

                    complete(res) {

                        console.log("存储程序运行了");

                    }

                });

            }

        })

    },

    //对好友框进行渲染

    Create\_User\_Block: function(user, index) {

        var New\_Friend\_Show = cc.instantiate(this.Friend\_Show);

        this.Friend\_Show\_Label.addChild(New\_Friend\_Show);

        //设置昵称

        console.log("设置昵称");

        New\_Friend\_Show.getChildByName("User\_Name").getComponent(cc.Label).string = user.nickname || user.nickName;

        //设置分数

        console.log("设置分数");

        console.log("此时的user为", user.KVDataList[0].value);

        New\_Friend\_Show.getChildByName("Best\_Score\_Text").getComponent(cc.Label).string = "" + user.KVDataList[0].value +

            "分";

        New\_Friend\_Show.getChildByName("Game\_Rank\_Show").getComponent(cc.Label).string = "" + (1 + index);

        //设置头像

        console.log("设置头像");

        cc.loader.load({

            url: user.avatarUrl,

            type: 'png'

        }, (err, texture) => {

            if (err) {

                console.error(err);

            }

            New\_Friend\_Show.getChildByName("Head\_Image\_Mask").getChildByName("Head\_Image").getComponent(cc.Sprite).spriteFrame =

                new cc.SpriteFrame(texture);

        })

    },

    // update (dt) {},

});

# 4.商店与角色模块

## 4.1商店

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 加载角色预览 | Loading\_Character |
| 2 | 角色信息 | Character\_Information.js |
| 3 | 角色购买 | Buy\_Character\_Sure |

### 4.1.1 加载角色预览

// Learn cc.Class:

//  - https://docs.cocos.com/creator/manual/en/scripting/class.html

// Learn Attribute:

//  - https://docs.cocos.com/creator/manual/en/scripting/reference/attributes.html

// Learn life-cycle callbacks:

//  - https://docs.cocos.com/creator/manual/en/scripting/life-cycle-callbacks.html

const Shop\_Character\_Local\_Varible = require('../Local\_Variible/Shop\_Character\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Shop\_Character\_Label:{

            default:null,

            type:cc.Prefab,

            serialzable:true,

        },

        Shop\_Character\_View:{

            default:null,

            type:cc.Node,

            serialzable:true,

        },

    },

    // LIFE-CYCLE CALLBACKS:

     onLoad: function(){

        var self = this;

        WeChat.Loading\_Shop\_Character();

        //Shop\_Character\_Local\_Varible.Shop\_Character\_User.length

        for(var i=0;i<Shop\_Character\_Local\_Varible.Shop\_Character\_User.length;i++){

            var New\_Shop\_Character\_Label=cc.instantiate(this.Shop\_Character\_Label);

            this.Shop\_Character\_View.addChild(New\_Shop\_Character\_Label);

            this.Loading\_Image(New\_Shop\_Character\_Label,Shop\_Character\_Local\_Varible.Shop\_Character\_User[i].Character\_Head\_Image);

            New\_Shop\_Character\_Label.getChildByName("Character\_Name").getComponent(cc.Label).string=""+Shop\_Character\_Local\_Varible.Shop\_Character\_User[i].Character\_Name;

        }

     },

    start () {

    },

    update (dt) {},

    Loading\_Image(self,Image\_Path){

        let \_url=Image\_Path;

        cc.loader.load({

            url:\_url,

            type:'jpg'

        },function(err,texture,test){

            var frame=new cc.SpriteFrame(texture);

            if(err){

                console.log("图片错误",err);

            }

            self.getChildByName("sprite").getComponent(cc.Sprite).spriteFrame=frame;

        })

}

});

### 4.1.2 角色信息

var Shop\_Character\_Local\_Varible = require('../Local\_Variible/Shop\_Character\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Buy\_Character\_Background:{

            default:null,

            type:cc.Prefab,

            serialzable:true,

        },

        Character\_Name:{

            default:null,

            type:cc.Label,

            serialzable:true,

        },

        Canvas:{

            default:null,

            type:cc.Node,

            serialzable:true,

        }

    },

    // LIFE-CYCLE CALLBACKS:

    on\_btn\_click: function() {

        var self = this;

        var This\_Name=this.Character\_Name.getComponent(cc.Label).string;

        WeChat.Loading\_Shop\_Character();

        for(var i=0;i<Shop\_Character\_Local\_Varible.Shop\_Character\_User.length;i++){

            var List\_Name=Shop\_Character\_Local\_Varible.Shop\_Character\_User[i].Character\_Name;

            if(This\_Name===List\_Name){

                //当前点击的角色

                var This\_information=Shop\_Character\_Local\_Varible.Shop\_Character\_User[i];

                var New\_Buy\_Character\_Background = cc.instantiate(this.Buy\_Character\_Background);

                this.Canvas.parent.parent.parent.addChild(New\_Buy\_Character\_Background);

                New\_Buy\_Character\_Background.setPosition(0,-300);

                console.log("图片地址为",This\_information);

                this.Loading\_Image(New\_Buy\_Character\_Background,This\_information.Character\_Head\_Image);

                New\_Buy\_Character\_Background.getChildByName("Character\_Name").getComponent(cc.Label).string=""+This\_information.Character\_Name;

                New\_Buy\_Character\_Background.getChildByName("Bounce\_Power\_Number\_Label").getComponent(cc.Label).string=""+This\_information.Character\_Jump\_Speed;

                New\_Buy\_Character\_Background.getChildByName("Weight\_Number\_Label").getComponent(cc.Label).string=""+This\_information.Character\_Fall\_Speed;

                New\_Buy\_Character\_Background.getChildByName("Speed\_Number\_Label").getComponent(cc.Label).string=""+This\_information.Character\_Fly\_Speed;

                New\_Buy\_Character\_Background.getChildByName("Skill\_Name\_Label").getComponent(cc.Label).string=""+This\_information.Skill\_Name;

                New\_Buy\_Character\_Background.getChildByName("Skill\_Effect\_Label").getComponent(cc.Label).string=""+This\_information.Skill\_Synopsis;

                New\_Buy\_Character\_Background.getChildByName("Price\_Label").getComponent(cc.Label).string=""+This\_information.Character\_Price;

                New\_Buy\_Character\_Background.getChildByName("Character\_Id").getComponent(cc.Label).string=""+This\_information.Character\_Id;

                break;

            }

        }

    },

    update (dt) {},

    Loading\_Image(self,Image\_Path){

        let \_url=Image\_Path;

        cc.loader.load({

            url:\_url,

            type:'jpg'

        },function(err,texture,test){

            var frame=new cc.SpriteFrame(texture);

            if(err){

                console.log("图片错误",err);

            }

            self.getChildByName("Character\_Image").getComponent(cc.Sprite).spriteFrame=frame;

        })

}

});

### 4.1.3 角色购买

var Shop\_Character\_Local\_Varible = require('../Local\_Variible/Shop\_Character\_Local\_Varible');

const User\_Have\_Character\_Local\_Varible = require('../Local\_Variible/User\_Have\_Character\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Reminder\_Box:{

            default:null,

            type:cc.Prefab,

            serialzable:true,

        },

        Character\_Id:{

            default:null,

            type:cc.Label,

            serialzable:true,

        },

        Price\_Label:{

            default:null,

            type:cc.Label,

            serialzable:true,

        },

        Canvas:{

            default:null,

            type:cc.Node,

            serialzable:true,

        }

    },

    // LIFE-CYCLE CALLBACKS:

    on\_btn\_click:function () {

        WeChat.Loading\_Shop\_Character();

        //读取当前点击的角色金币

        var This\_Information\_Price=this.Price\_Label.getComponent(cc.Label).string;

        //读取当前点击的角色信息

        var This\_Information=this.Character\_Id.getComponent(cc.Label).string;

        //读取用户拥有的金币

        var User\_Gold=Global\_Variable.Gold;

        var New\_Reminder\_Box = cc.instantiate(this.Reminder\_Box);

        this.Canvas.parent.parent.addChild(New\_Reminder\_Box);

        New\_Reminder\_Box.setPosition(0,-400);

        var flag=0;

        for(var i=0;i<User\_Have\_Character\_Local\_Varible.User\_Have\_Character.length;i++){

        //读取用户拥有的角色信息

        var User\_Character\_Information=User\_Have\_Character\_Local\_Varible.User\_Have\_Character[i];

            //判断已经拥有角色

            //判断是否是本机用户

                if((This\_Information==User\_Character\_Information.Character\_Id)&&(Global\_Variable.openid==User\_Character\_Information.openid)){

                    New\_Reminder\_Box.getChildByName("Reminder\_Text").getComponent(cc.Label).string="您已拥有该商品";

                    flag=1;

                    break;

                    //输出已经拥有

        }

        }

        console.log("小鸟价格",This\_Information\_Price);

        console.log("用户金币数",User\_Gold);

        //判断用户金币是否购买的起当前小鸟

        if((User\_Gold<This\_Information\_Price)&&(flag==0)){

            New\_Reminder\_Box.getChildByName("Reminder\_Text").getComponent(cc.Label).string="金币不足购买商品";

        }

        if((User\_Gold>=This\_Information\_Price)&&(flag==0)){

            New\_Reminder\_Box.getChildByName("Reminder\_Text").getComponent(cc.Label).string="购买成功该商品";

            User\_Gold=User\_Gold-This\_Information\_Price;

            WeChat.Buy\_Character\_Update(User\_Gold,This\_Information);

        }

    },

        update (dt) {},

});

## 4.2 角色

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 加载角色预览 | Loading\_Role |
| 2 | 拥有角色信息 | Have\_Character\_Information |
| 3 | 更换角色 | Change\_Character |

### 4.2.1 加载角色预览

// Learn cc.Class:

//  - https://docs.cocos.com/creator/manual/en/scripting/class.html

// Learn Attribute:

//  - https://docs.cocos.com/creator/manual/en/scripting/reference/attributes.html

// Learn life-cycle callbacks:

//  - https://docs.cocos.com/creator/manual/en/scripting/life-cycle-callbacks.html

const User\_Have\_Character\_Local\_Varible = require('../Local\_Variible/User\_Have\_Character\_Local\_Varible');

var Shop\_Character\_Local\_Varible = require('../Local\_Variible/Shop\_Character\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Character\_Label:{

            default:null,

            type:cc.Prefab,

            serialzable:true,

        },

        Character\_View:{

            default:null,

            type:cc.Node,

            serialzable:true,

        },

    },

    // LIFE-CYCLE CALLBACKS:

     onLoad: function(){

        var self = this;

        WeChat.Loading\_Character();

        //遍历User\_Character\_Information的表

        for(var i=0;i<User\_Have\_Character\_Local\_Varible.User\_Have\_Character.length;i++){

            //User\_Character\_Information放的是只有2个记录的小表(User\_Have\_Character)的信息

            var User\_Character\_Information=User\_Have\_Character\_Local\_Varible.User\_Have\_Character[i];

            //User\_Character\_Number放的是Character的序号

            var User\_Character\_Number=User\_Have\_Character\_Local\_Varible.User\_Have\_Character[i].Character\_Id;

            //判断用户是否有该小鸟

            //Global\_Variable.openid是当前用户的openid

            if(Global\_Variable.openid==User\_Character\_Information.openid){

                // console.log("角色长度",User\_Have\_Character\_Local\_Varible.User\_Have\_Character.length);

                for(var j=0;j<Shop\_Character\_Local\_Varible.Shop\_Character\_User.length;j++){

                    if(User\_Character\_Number==Shop\_Character\_Local\_Varible.Shop\_Character\_User[j].Character\_Id){

                        var New\_Character\_Label=cc.instantiate(this.Character\_Label);

                        this.Character\_View.addChild(New\_Character\_Label);

                        this.Loading\_Image(New\_Character\_Label,Shop\_Character\_Local\_Varible.Shop\_Character\_User[j].Character\_Head\_Image);

                        New\_Character\_Label.getChildByName("Character\_Name").getComponent(cc.Label).string=""+Shop\_Character\_Local\_Varible.Shop\_Character\_User[j].Character\_Name;

                    }

                }

            }

        }

     },

    start () {

    },

    update (dt) {},

    Loading\_Image(self,Image\_Path){

        let \_url=Image\_Path;

        cc.loader.load({

            url:\_url,

            type:'jpg'

        },function(err,texture,test){

            var frame=new cc.SpriteFrame(texture);

            if(err){

                console.log("图片错误",err);

            }

            self.getChildByName("sprite").getComponent(cc.Sprite).spriteFrame=frame;

        })

}

});

### 4.2.2 拥有角色信息

// Learn cc.Class:

//  - https://docs.cocos.com/creator/manual/en/scripting/class.html

// Learn Attribute:

//  - https://docs.cocos.com/creator/manual/en/scripting/reference/attributes.html

// Learn life-cycle callbacks:

//  - https://docs.cocos.com/creator/manual/en/scripting/life-cycle-callbacks.html

var Shop\_Character\_Local\_Varible = require('../Local\_Variible/Shop\_Character\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Buy\_Character\_Background:{

            default:null,

            type:cc.Prefab,

            serialzable:true,

        },

        Character\_Name:{

            default:null,

            type:cc.Label,

            serialzable:true,

        },

        Canvas:{

            default:null,

            type:cc.Node,

            serialzable:true,

        }

    },

    // LIFE-CYCLE CALLBACKS:

    on\_btn\_click: function() {

        var self = this;

        var This\_Name=this.Character\_Name.getComponent(cc.Label).string;

        WeChat.Loading\_Shop\_Character();

        for(var i=0;i<Shop\_Character\_Local\_Varible.Shop\_Character\_User.length;i++){

            var List\_Name=Shop\_Character\_Local\_Varible.Shop\_Character\_User[i].Character\_Name;

            if(This\_Name===List\_Name){

                //当前点击的角色

                var This\_information=Shop\_Character\_Local\_Varible.Shop\_Character\_User[i];

                var New\_Buy\_Character\_Background = cc.instantiate(this.Buy\_Character\_Background);

                this.Canvas.parent.parent.parent.addChild(New\_Buy\_Character\_Background);

                New\_Buy\_Character\_Background.setPosition(0,-210);

                console.log("图片地址为",This\_information);

                this.Loading\_Image(New\_Buy\_Character\_Background,This\_information.Character\_Head\_Image);

                New\_Buy\_Character\_Background.getChildByName("Character\_Name").getComponent(cc.Label).string=""+This\_information.Character\_Name;

                New\_Buy\_Character\_Background.getChildByName("Bounce\_Power\_Number\_Label").getComponent(cc.Label).string=""+This\_information.Character\_Jump\_Speed;

                New\_Buy\_Character\_Background.getChildByName("Weight\_Number\_Label").getComponent(cc.Label).string=""+This\_information.Character\_Fall\_Speed;

                New\_Buy\_Character\_Background.getChildByName("Speed\_Number\_Label").getComponent(cc.Label).string=""+This\_information.Character\_Fly\_Speed;

                New\_Buy\_Character\_Background.getChildByName("Skill\_Name\_Label").getComponent(cc.Label).string=""+This\_information.Skill\_Name;

                New\_Buy\_Character\_Background.getChildByName("Skill\_Effect\_Label").getComponent(cc.Label).string=""+This\_information.Skill\_Synopsis;

               New\_Buy\_Character\_Background.getChildByName("Character\_Id").getComponent(cc.Label).string=""+This\_information.Character\_Id;

                break;

            }

        }

    },

    update (dt) {},

    Loading\_Image(self,Image\_Path){

        let \_url=Image\_Path;

        cc.loader.load({

            url:\_url,

            type:'jpg'

        },function(err,texture,test){

            var frame=new cc.SpriteFrame(texture);

            if(err){

                console.log("图片错误",err);

            }

            self.getChildByName("Character\_Image").getComponent(cc.Sprite).spriteFrame=frame;

        })

}

});

### 4.2.3 更换角色

// Learn cc.Class:

//  - https://docs.cocos.com/creator/manual/en/scripting/class.html

// Learn Attribute:

//  - https://docs.cocos.com/creator/manual/en/scripting/reference/attributes.html

// Learn life-cycle callbacks:

//  - https://docs.cocos.com/creator/manual/en/scripting/life-cycle-callbacks.html

var Shop\_Character\_Local\_Varible = require('../Local\_Variible/Shop\_Character\_Local\_Varible');

cc.Class({

    extends: cc.Component,

    properties: {

        Canvas:{

            default:null,

            type:cc.Node,

            serialzable:true,

        }

    },

    // LIFE-CYCLE CALLBACKS:

    on\_btn\_click: function() {

        console.log("角色ID为",this.Canvas.getChildByName("Character\_Id").getComponent(cc.Label).string);

        var Current\_id=this.node.getChildByName("Character\_Id").getComponent(cc.Label).string;

        WeChat.Updating\_Current\_Character\_id(Current\_id);

        this.Canvas.destroy();

    },

    update (dt) {},

});

# 5.排行榜模块

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 加载世界排行榜 | Loading\_Rank |
| 2 | 加载玩家成绩信息 | My\_Rank |
| 3 | 取消举报 | Report\_Cancel |
| 4 | 确定举报 | Report\_Confirm |
| 5 | 举报界面 | Report\_User |

## 5.1 加载世界排行榜

//下载世界排名列表

var Rank\_Local\_Varible = require('Rank\_Local\_Variable');

cc.Class({

    extends: cc.Component,

    properties: {

    Rank\_User\_Label:{

                default:null,

                type:cc.Prefab,

                serialzable:true,

     },//玩家框

     Rank\_View:{

                default:null,

                type:cc.Node,

                serialzable:true,

     },//排名框

    },

    onLoad:function () {

        var self = this;

        WeChat.Loading\_World\_Rank();

        console.log(Rank\_Local\_Varible.Word\_Rank\_User);

        //Rank\_Local\_Varible.Word\_Rank\_User.length

        //循环输出玩家框

        for(var i=0;i<Rank\_Local\_Varible.Word\_Rank\_User.length;i++){

            var New\_Rank\_User\_Label = cc.instantiate(this.Rank\_User\_Label);

            this.Rank\_View.addChild(New\_Rank\_User\_Label);

            this.Loading\_Image(New\_Rank\_User\_Label,Rank\_Local\_Varible.Word\_Rank\_User[i].Head\_Iamge);

            New\_Rank\_User\_Label.getChildByName("Game\_Rank\_Show").getComponent(cc.Label).string=""+(i+1);

            New\_Rank\_User\_Label.getChildByName("User\_Name").getComponent(cc.Label).string=""+Rank\_Local\_Varible.Word\_Rank\_User[i].User\_Name;

            New\_Rank\_User\_Label.getChildByName("Best\_Score\_Text").getComponent(cc.Label).string=""+Rank\_Local\_Varible.Word\_Rank\_User[i].Best\_Score+"分";

        }

    },

    start () {

    },

    update (dt) {},

    Loading\_Image(self,Image\_Path){

                let \_url=Image\_Path;

                cc.loader.load({

                    url:\_url,

                    type:'jpg'

                },function(err,texture,test){

                    var frame=new cc.SpriteFrame(texture);

                    if(err){

                        console.log("图片错误",err);

                    }

                    self.getChildByName("Head\_Image\_Mask").getChildByName("Head\_Image").getComponent(cc.Sprite).spriteFrame=frame;

                })

    }

});

## 5.2加载玩家成绩信息

//加载自己的成绩信息

cc.Class({

    extends: cc.Component,

    properties: {

        User\_Heading\_Image:{

                    default:null,

                    type:cc.Sprite,

                    serialzable:true,

        },//使用者头像

        User\_Name:{

                default:null,

                    type:cc.Label,

                    serialzable:true,

        },//使用者名字

        Best\_Scroe\_Text:{

                    default:null,

                    type:cc.Label,

                    serialzable:true,

        },

    },

    //加载自己的信息

     onLoad: function(){

         this.Loading\_Image(this,Global\_Variable.User\_Head\_Image);

         this.User\_Name.string=Global\_Variable.User\_Name;

         this.Best\_Scroe\_Text.string=Global\_Variable.Best\_Score+"分";

     },

    start () {

    },

    // update (dt) {},

    //加载图片

    Loading\_Image(self,Image\_Path){

                let \_url=Image\_Path;

                cc.loader.load({

                    url:\_url,

                    type:'jpg'

                },function(err,texture,test){

                    var frame=new cc.SpriteFrame(texture);

                    if(err){

                        console.log("图片错误",err);

                    }

                    self.User\_Heading\_Image.getComponent(cc.Sprite).spriteFrame=frame;

                })

    }

});

## 5.3 取消举报

//取消会议记录

cc.Class({

    extends: cc.Component,

    properties: {

    },

    // LIFE-CYCLE CALLBACKS:

    // onLoad () {},

    start () {

    },

    on\_btn\_click: function() {

        this.node.destroy();

    }

    // update (dt) {},

});

## 5.4确定举报

//确定举报

var Report\_Local\_Variable = require('Report\_Local\_Variable');

cc.Class({

    extends: cc.Component,

    properties: {

        Report\_Content: {

            default: null,

            type: cc.Label,

            serialzable: true,

        },//举报框

    },

    // LIFE-CYCLE CALLBACKS:

    // onLoad () {},

    start() {

    },

    on\_btn\_click: function() {

        //载入举报人的信息

        var Reported\_User\_Openid = Report\_Local\_Variable.openid;

        var Report\_Text = this.Report\_Content.getComponent(cc.Label).string;

        console.log("被举报人OPENID", Reported\_User\_Openid, "举报内容为", Report\_Text);

        //上传举报信息

        WeChat.Uploading\_Reported\_Information(Report\_Local\_Variable.openid, Report\_Text);

        this.node.destroy();

    }

    // update (dt) {},

});

## 5.5 举报界面

//弹出举报框

var Report\_Local\_Variable = require('Report\_Local\_Variable');

var Rank\_Local\_Varible = require('Rank\_Local\_Variable');

cc.Class({

    extends: cc.Component,

    properties: {

        Report\_Label: {

            default: null,

            type: cc.Prefab,

            serialzable: true,

        }, //举报框

        Canvas: {

            default: null,

            type: cc.Node,

            serialzable: true,

        }, //玩家框节点

        Report\_Rank: {

            default: null,

            type: cc.Label,

            serialzable: true,

        }, //被举报玩家

    },

    start() {

    },

    on\_btn\_click: function() {

        //创建举报信息框

        var New\_Report\_Label = cc.instantiate(this.Report\_Label);

        this.Canvas.parent.parent.parent.addChild(New\_Report\_Label);

        New\_Report\_Label.setPosition(0, 0);

        //载入信息

        console.log("被举报人的名词", this.Report\_Rank.getComponent(cc.Label).string);

        var Rank\_Number = Number(this.Report\_Rank.getComponent(cc.Label).string) - 1;

        console.log("被举报人的openid", Report\_Local\_Variable.openid);

        Report\_Local\_Variable.openid = Rank\_Local\_Varible.Word\_Rank\_User[Rank\_Number].openid;

    }

    // update (dt) {},

});

# 6局部变量

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 游戏难度系数 | Game\_Difficulty\_Local\_Varible |
| 2 | 游戏界面 | Game\_Local\_Varible |
| 3 | 跳转难度界面 | Jump\_DIfficulty |
| 4 | 排行榜界面 | Rank\_Local\_Variable |
| 5 | 举报界面 | Report\_Local\_Variable |
| 6 | 商店角色界面 | Shop\_Character\_Local\_Varible |
| 7 | 玩家拥有角色界面 | User\_Have\_Character\_Local\_Varible |

## 6.1游戏难度系数

//游戏难度系数的局部变量

module.exports = {

    //游戏难度系数

    Difficulty\_Ratio: 1,

    //是否是困难模式

    Is\_Difficulty: false,

};

## 6.2游戏界面

//游戏界面的局部变量

module.exports = {

    //分数统计

    Fraction: 0,

    //获得金币

    Gold: 0,

    //当前地图

    Current\_Map: {

        default: null,

        type: cc.SpriteFrame,

        serialzable: true,

    },

};

## 6.3 跳转难度界面

//跳转难度界面

cc.Class({

    extends: cc.Component,

    properties: {

        Choose\_Difficulty: {

            default: null,

            type: cc.Sprite,

            serialzable: true,

        }, //难度框

        Jump\_Jump: {

            default: null,

            type: cc.Sprite,

            serialzable: true,

        }, //点击图片

    },

    onLoad: function() {

        this.Choose\_Difficulty.node.active = false;

        // 初始化跳跃动作

        this.node.on(cc.Node.EventType.TOUCH\_START, this.onTouchMove, this);

    },

    onDestroy() {

        // 取消键盘输入监听

    },

    start() {

    },

    update: function(dt) {

    },

    onTouchMove(event) {

        //管理Jump图片

        this.Jump\_Jump.node.active = false;

        //打开难度选择框

        this.Choose\_Difficulty.node.active = true;

    },

});

## 6.4 排行榜界面

module.exports = {

    //�������������Ϣ

    Word\_Rank\_User: null,

};

## 6.5举报界面

//举报

module.exports = {

    //举报列表

    openid:"",

};

## 6.6商店角色界面

module.exports = {

    Shop\_Character\_User: null,

};

## 6.7用户拥有角色界面

module.exports = {

    User\_Have\_Character: null,

};

# 7.云函数

|  |  |  |
| --- | --- | --- |
| 源代码清单概述 | | |
|  | | |
| No | 模块名称 | 程序名 |
| 1 | 角色 | Character |
| 2 | 游戏 | Game\_Settlement |
| 3 | 加载小鸟图片 | Loading\_Bird\_Image |
| 4 | 加载游戏 | Loading\_Game |
| 5 | 加载世界排行榜 | Loading\_World\_Rank |
| 6 | 商店 | Shop |
| 7 | 商店购买 | Shop\_Buy |

## 7.1角色

// 云函数入口文件

const cloud = require('wx-server-sdk')

cloud.init()

const db=cloud.database();

const Character\_Information\_Table=db.collection('Character');

const User\_Have\_Character\_Information\_Table=db.collection('User\_Have\_Character');

// 云函数入口函数

exports.main = async (event, context) => {

  const wxContext = cloud.getWXContext()

   let Character\_Information=await Character\_Information\_Table.get();

   let User\_Have\_Character\_Information=await User\_Have\_Character\_Information\_Table.get();

  return {

    event,

    openid: wxContext.OPENID,

    appid: wxContext.APPID,

    unionid: wxContext.UNIONID,

    Character\_Information:Character\_Information,

    User\_Have\_Character\_Information:User\_Have\_Character\_Information,

  }

}

## 7.2游戏

// 云函数入口文件

const cloud = require('wx-server-sdk')

cloud.init()

const db=cloud.database();

const User\_Information\_Table=db.collection('User');

const Records\_Information\_Table=db.collection('Records');

var myDate = new Date();

const \_=db.command;

// 云函数入口函数

exports.main = async (event, context) => {

  const wxContext = cloud.getWXContext()

  let User\_Information=await User\_Information\_Table.where({

    openid:wxContext.OPENID

  }).get();

  if(User\_Information.data[0].Compassion<=0){

    if(event.Score>Best\_Score){

      await User\_Information\_Table.where({

        openid:wxContext.OPENID

      }).update({

        data:{

          Best\_Score:event.Score

        },

    })

    }

    return {

      openid: wxContext.OPENID,

      appid: wxContext.APPID,

      unionid: wxContext.UNIONID,

    }

  }

  var Current\_Gold = User\_Information.data[0].Gold+event.Add\_Gold;

  var Best\_Score=User\_Information.data[0].Best\_Score;

  var Current\_Character\_Id = User\_Information.data[0].Current\_Character\_Id;

  if(event.Score>Best\_Score){

    await User\_Information\_Table.where({

      openid:wxContext.OPENID

    }).update({

      data:{

        Gold:Current\_Gold,

        Best\_Score:event.Score,

        Compassion:\_.inc(-1)

      },

  })

  }else{

    await User\_Information\_Table.where({

      openid:wxContext.OPENID

    }).update({

      data:{

        Gold:Current\_Gold,

        Compassion:\_.inc(-1)

      },

  })

  }

  console.log(User\_Information)

  let addData={

    openid:wxContext.OPENID,

    Records\_Score: event.Score,

    Gold:Current\_Gold,

    Character\_Id:Current\_Character\_Id,

    Records\_Time:db.serverDate(),

  }

  console.log(addData)

  \_isAdd=Records\_Information\_Table.add({

    data:addData

  })

  console.log(\_isAdd)

  return {

    openid: wxContext.OPENID,

    appid: wxContext.APPID,

    unionid: wxContext.UNIONID,

    isAdd:\_isAdd,

    Gold:Current\_Gold,

  }

}

## 7.3加载小鸟图片

// 云函数入口文件

const cloud = require('wx-server-sdk')

cloud.init()

const db=cloud.database();

const Character\_Information\_Table=db.collection('Character');

// 云函数入口函数

exports.main = async (event, context) => {

  const wxContext = cloud.getWXContext()

  let Character\_Information=await Character\_Information\_Table.where({

    Character\_Id:event.Character\_Id

  }).get();

  Character\_Image\_1=Character\_Information.data[0].Character\_Image\_1

  Character\_Image\_2=Character\_Information.data[0].Character\_Image\_2

  Character\_Image\_3=Character\_Information.data[0].Character\_Image\_3

  Character\_Image\_4=Character\_Information.data[0].Character\_Image\_4

  return {

    event,

    openid: wxContext.OPENID,

    appid: wxContext.APPID,

    unionid: wxContext.UNIONID,

    Character\_Image1:Character\_Image\_1,

    Character\_Image2:Character\_Image\_2,

    Character\_Image3:Character\_Image\_1,

    Character\_Image4:Character\_Image\_2,

  }

}

## 7.4加载游戏

// 云函数入口文件

const cloud = require('wx-server-sdk')

cloud.init()

const db=cloud.database();

const User\_Information\_Table=db.collection('User');

// 云函数入口函数

exports.main = async (event, context) => {

  const wxContext = cloud.getWXContext()

  let User\_Information=await User\_Information\_Table.where({

    openid:wxContext.OPENID

  }).get();

  var Gold = User\_Information\_Table.Gold;

  var  Diamond= User\_Information\_Table.Diamond;

  var  Compassion= User\_Information\_Table.Compassion;

  console.log(User\_Information)

  return {

    openid: wxContext.OPENID,

    appid: wxContext.APPID,

    unionid: wxContext.UNIONID,

    Gold:Gold,

    Diamond: Diamond,

    Compassion:Compassion,

  }

}

## 7.5加载世界排行榜

// 云函数入口文件

const cloud = require('wx-server-sdk')

cloud.init()

const db=cloud.database();

const User\_Information\_Table=db.collection('User');

// 云函数入口函数

exports.main = async (event, context) => {

  const wxContext = cloud.getWXContext()

  let User\_Information=await User\_Information\_Table.orderBy("Best\_Score",'desc').limit(10).get();

  return {

    event,

    openid: wxContext.OPENID,

    appid: wxContext.APPID,

    unionid: wxContext.UNIONID,

    User\_Information:User\_Information,

  }

}

## 7.6商店

// 云函数入口文件

const cloud = require('wx-server-sdk')

cloud.init()

const db=cloud.database();

const Character\_Information\_Table=db.collection('Character');

const User\_Have\_Character\_Information\_Table=db.collection('User\_Have\_Character');

// 云函数入口函数

exports.main = async (event, context) => {

  const wxContext = cloud.getWXContext()

   let Character\_Information=await Character\_Information\_Table.get();

   let User\_Have\_Character\_Information=await User\_Have\_Character\_Information\_Table.get();

    return {

      event:event,

      openid: wxContext.OPENID,

      appid: wxContext.APPID,

      unionid: wxContext.UNIONID,

      Character\_Information:Character\_Information,

      User\_Have\_Character\_Information:User\_Have\_Character\_Information,

    }

}

## 7.7商店购买

// 云函数入口文件

const cloud = require('wx-server-sdk')

cloud.init()

const db=cloud.database();

const User\_Information\_Table=db.collection('User');

const User\_Have\_Character\_Information\_Table=db.collection('User\_Have\_Character');

// 云函数入口函数

exports.main = async (event, context) => {

  const wxContext = cloud.getWXContext()

  let User\_Information=await User\_Information\_Table.get();

  let User\_Have\_Character\_Information=await User\_Have\_Character\_Information\_Table.get();

  console.log(User\_Have\_Character\_Information)

  let addData={

    Character\_Id:event.Update\_Character,

    openid:wxContext.OPENID,

  }

  \_isAdd=User\_Have\_Character\_Information\_Table.add({

    data:addData,

  })

  await User\_Information\_Table.where({

    openid:wxContext.OPENID

  }).update({

    data:{

      Gold:event.Update\_Gold,

    },

})

  return {

    event,

    openid: wxContext.OPENID,

    appid: wxContext.APPID,

    unionid: wxContext.UNIONID,

  }

}