var UM:UserManager,TM:TimeManager,EM:EventManager

class Main extends eui.UILayer {

private loadingView: MainLoadingUI;

protected createChildren(): void {

super.createChildren();

console.log('\_10')

//注入自定义的素材解析器

var assetAdapter = new AssetAdapter();

this.stage.registerImplementation("eui.IAssetAdapter",assetAdapter);

this.stage.registerImplementation("eui.IThemeAdapter",new ThemeAdapter());

this.setScaleMode();

//设置加载进度界面

this.loadingView = MainLoadingUI.getInstance();

RES.addEventListener(RES.ResourceEvent.CONFIG\_COMPLETE, this.onConfigComplete, this);

RES.loadConfig("resource/default.res.json", "resource/");

UM = UserManager.getInstance();

TM = TimeManager.getInstance();

EM = EventManager.getInstance();

Config.initURLRequest();

console.log('\_1a')

}

private setScaleMode(){

this.stage.scaleMode = egret.StageScaleMode.FIXED\_WIDTH;

}

private onConfigComplete(event:RES.ResourceEvent):void {

console.log('\_1b')

RES.removeEventListener(RES.ResourceEvent.CONFIG\_COMPLETE, this.onConfigComplete, this);

var theme = new eui.Theme("resource/default.thm.json", this.stage);

theme.addEventListener(eui.UIEvent.COMPLETE, this.onThemeLoadComplete, this);

RES.addEventListener(RES.ResourceEvent.GROUP\_COMPLETE, this.onResourceLoadComplete, this);

RES.addEventListener(RES.ResourceEvent.GROUP\_LOAD\_ERROR, this.onResourceLoadError, this);

RES.addEventListener(RES.ResourceEvent.GROUP\_PROGRESS, this.onResourceProgress, this);

RES.addEventListener(RES.ResourceEvent.ITEM\_LOAD\_ERROR, this.onItemLoadError, this);

RES.loadGroup("game");

}

private isThemeLoadEnd: boolean = false;

private onThemeLoadComplete(): void {

this.isThemeLoadEnd = true;

console.log('\_1c')

this.createScene();

}

private isResourceLoadEnd: boolean = false;

private onResourceLoadComplete(event:RES.ResourceEvent):void {

console.log('\_1d')

if (event.groupName == "game") {

this.isResourceLoadEnd = true;

this.removeLoadEvent();

this.createScene();

}

}

private removeLoadEvent(){

this.loadingView.setFinish();

RES.removeEventListener(RES.ResourceEvent.GROUP\_COMPLETE, this.onResourceLoadComplete, this);

RES.removeEventListener(RES.ResourceEvent.GROUP\_LOAD\_ERROR, this.onResourceLoadError, this);

RES.removeEventListener(RES.ResourceEvent.GROUP\_PROGRESS, this.onResourceProgress, this);

RES.removeEventListener(RES.ResourceEvent.ITEM\_LOAD\_ERROR, this.onItemLoadError, this);

}

private createScene(){

if(this.isThemeLoadEnd && this.isResourceLoadEnd){

this.startCreateScene();

}

}

private onItemLoadError(event:RES.ResourceEvent):void {

console.warn("Url:" + event.resItem.url + " has failed to load");

}

/\*\*

\* 资源组加载出错

\* Resource group loading failed

\*/

private onResourceLoadError(event:RES.ResourceEvent):void {

console.warn("Group:" + event.groupName + " has failed to load");

this.onResourceLoadComplete(event);

}

/\*\*

\* preload资源组加载进度

\* loading process of preload resource

\*/

private onResourceProgress(event:RES.ResourceEvent):void {

console.log(event.itemsLoaded, event.itemsTotal)

if (event.groupName == "game") {

this.loadingView.setProgress(event.itemsLoaded, event.itemsTotal);

}

}

/\*\*

\* 创建场景界面

\* Create scene interface

\*/

protected startCreateScene(): void {

GameManager.stage = this.stage;

GameManager.container = this;

GameManager.getInstance().init();

console.log('\_11')

var wx = window['wx'];

if(!wx)

{

GameUI.getInstance().show();

return;

}

console.log('\_12')

UserManager.getInstance().getUserInfo(()=>{

GameUI.getInstance().show();

});

}

}

class CarManager {

private static instance:CarManager;

public static getInstance():CarManager {

if (!this.instance)

this.instance = new CarManager();

return this.instance;

}

public constructor() {

this.maxLevel = ObjectUtil.objLength(this.levelData)

}

public maxLevel = \_get['level'] || 0;

public skinid = 1;

public levelData = {1:20000}

public skins = [1,2]

public skinsData:any = {};

public initData(data){

this.skinid = data.skinid;

this.levelData = data.levelData;

this.skins = data.skins;

this.skinsData = data.skinsData || {};

this.maxLevel = ObjectUtil.objLength(this.levelData)

}

public getLevelStar(level){

var cd = this.getLevelCD(level);

if(cd)

return GameData.getInstance().getStarByLevel(level,cd)

return 0;

}

public getAllStar(){

var star = 0;

for(var s in this.levelData)

{

star += this.getLevelStar(s);

}

return star;

}

public getLevelCD(level){

return this.levelData[level] || 0

}

public isHaveSkin(skinid){

return this.skins.indexOf(skinid) != -1;

}

public getSkinValue(skinid){

var carVO = GameData.getInstance().carData[skinid];

switch(carVO.buyType)

{

case 0://默认

break

case 1:

//this.rateText.text = '完成第'+max+'关'

return this.maxLevel

case 2:

//this.rateText.text = max + '小时后获得'

return (TM.now() - (this.skinsData.time || 0))/3600

case 3:

//this.rateText.text = '邀请'+max+'个新用户'

return ObjectUtil.objLength(this.skinsData[skinid] || {});

case 4:

//this.rateText.text = '观看广告'+max+'次'

return this.skinsData[skinid] || 0;

}

return 0;

}

public sendSuccess(useTime,fun?){

var GD = GameData.getInstance();

var lastCD = this.getLevelCD(GD.level);

SharedObjectManager.getInstance().removeMyValue('chapter'+GD.level)

this.upWXData();

if(!lastCD || useTime < lastCD)

{

//var newLevelData = ObjectUtil.clone(this.levelData)

var newLevelData = {};

newLevelData[GD.level] = useTime;

var lastStar = this.getAllStar();

WXDB.updata('user',{levelData:newLevelData},()=>{

this.levelData[GD.level] = useTime;

this.maxLevel = ObjectUtil.objLength(this.levelData)

if(lastStar != this.getAllStar())

this.upWXData();

fun && fun();

})

return;

}

fun && fun();

}

public setCarSkin(skinid,fun?){

this.skinid = skinid;

GameUI.getInstance().renewCar();

WXDB.updata('user',{skinid:skinid},()=>{

fun && fun();

})

}

public buySkin(skinid,fun?){

var newSkins = this.skins.concat(skinid);

var t = TM.now();

WXDB.updata('user',{skins:newSkins,skinsData:{time:t}},()=>{

this.skins.push(skinid);

this.skinsData.time = t

EM.dispatch(GameEvent.client.SKIN\_CHANGE)

fun && fun();

})

}

public upWXData(){

var wx = window['wx'];

if(!wx)

return;

wx.setUserCloudStorage({

KVDataList: [{ key: 'score', value: this.getAllStar() + ',' + TM.now()}],

success: res => {

console.log(res);

},

fail: res => {

console.log(res);

}

});

}

}

class CarMC extends eui.Group{

private static pool = [];

public static createItem():CarMC {

var item:CarMC = this.pool.pop();

if (!item) {

item = new CarMC();

}

return item;

}

public static freeItem(item) {

if (!item)

return;

item.remove();

this.pool.push(item);

}

public speed;

private mc:eui.Image;

public constructor() {

super();

this.mc = new eui.Image('car\_png');

this.addChild(this.mc);

this.mc.x = this.mc.y = 0;

this.width = 238

this.height = 320

}

public remove(){

MyTool.removeMC(this);

}

public setCar(skinid):void{

var index = GameData.getInstance().carData[skinid].artid;

var w = this.width

var h = this.height

var x = (index-1)%4

var y = Math.floor((index-1)/4)

this.mc.scrollRect = new egret.Rectangle(x\*w,y\*h,w,h)

}

}

class GameData {

private static instance:GameData;

public static getInstance():GameData {

if (!this.instance)

this.instance = new GameData();

return this.instance;

}

public static speedRate = 0.25; //速度转像素的比例

public static Frame = 60; //帧频

public static MPV = 3600/1000 \* 60 \* 0.25; //米与象素转换的常量 =54

public static MaxLevel = 30; //关卡总数

public static MaxCar = 12; //关卡总数

public static AlertMeter = 100; //警告米数

public static FailDecTime = 10\*1000; //碰到红线扣除的时间

public isPlaying = false;

public speed = 0;

public maxSpeed = 0;

public carid = 0;

public addSpeed\_ = 0;

public decSpeed\_ = 0;

public addSpeedNum = 0; //调用加速的次数

public decSpeedNum = 0; //调用减速的次数

public lastAddSpeedTime = 0;

public maxLen = 0;

public startTime = 0;

public baseSpeed = 20;

public maxTime = 30\*1000;

public redArr = [];

public level = 1;

public countDown = 0;

public passMeter = 0;//玩家经过的米数

public carData = {

1:{maxSpeed:100,addSpeed:5,decSpeed:15,buyType:0,buyValue:0,artid:1},

2:{maxSpeed:110,addSpeed:6,decSpeed:15,buyType:1,buyValue:3,artid:2},

3:{maxSpeed:120,addSpeed:7,decSpeed:20,buyType:2,buyValue:1,artid:8},

4:{maxSpeed:125,addSpeed:8,decSpeed:25,buyType:3,buyValue:1,artid:11},

5:{maxSpeed:130,addSpeed:8,decSpeed:30,buyType:1,buyValue:10,artid:3},

6:{maxSpeed:135,addSpeed:9,decSpeed:35,buyType:2,buyValue:3,artid:4},

7:{maxSpeed:140,addSpeed:9,decSpeed:30,buyType:3,buyValue:2,artid:7},

8:{maxSpeed:145,addSpeed:10,decSpeed:35,buyType:2,buyValue:6,artid:6},

9:{maxSpeed:150,addSpeed:10,decSpeed:40,buyType:2,buyValue:12,artid:9},

10:{maxSpeed:155,addSpeed:11,decSpeed:35,buyType:1,buyValue:20,artid:10},

11:{maxSpeed:160,addSpeed:11,decSpeed:45,buyType:3,buyValue:3,artid:5},

12:{maxSpeed:160,addSpeed:12,decSpeed:50,buyType:2,buyValue:24,artid:12},

}

//米 转 象素

public meterToPix(m){

return m\*GameData.MPV

}

//象素 转 米

public pixToMeter(p){

return p/GameData.MPV

}

public getStarByLevel(lv,useTime){

var maxTime = this.getMaxTime(lv);

var rate = useTime/maxTime

if(rate < 0.7)

return 3;

if(rate < 0.85)

return 2;

return 1;

}

public getMaxTime(lv){

return Math.min((lv-1)\*5 + 25,180)\*1000

}

public onRunSpeed(){

if(!this.countDown)

{

if(this.lastAddSpeedTime)

{

var cd = egret.getTimer() - this.lastAddSpeedTime

this.passMeter += (this.speed\*1000/3600) \* (cd/1000)

}

this.lastAddSpeedTime = egret.getTimer()

}

}

public onGameStart(lv){

this.level = lv;

this.isPlaying = true;

this.speed = this.baseSpeed;

this.startTime = egret.getTimer();

this.passMeter = 0

this.countDown = 4;

this.lastAddSpeedTime = 0;

//生成关卡数据

this.maxTime = this.getMaxTime(this.level)

var maxLen = 100\*1000/3600\*(this.maxTime/1000);//100KM速度行使指定时间的距离（米）

maxLen = Math.floor(maxLen\*(0.5 + this.level/GameData.MaxLevel))//最终的行走距离

var addDec = 100 + GameData.AlertMeter - 100\*this.level/GameData.MaxLevel;//两个红块间的间隔

this.redArr = []

var currentPos = 0;

while(currentPos < maxLen)

{

currentPos += addDec + Math.random()\*addDec\*0.4 + 10;

var oo = {

start:currentPos,

speed:60 + Math.floor(Math.random()\*4)\*10 + Math.floor(this.level/8)\*10

}

this.redArr.push(oo)

}

this.redArr.pop();

this.maxLen = maxLen;

console.log(this.maxLen,this.redArr);

}

public addSpeed(isAdd){

if(isAdd)

{

this.addSpeedNum += 1;

if(this.addSpeedNum > 30)

this.addSpeedNum = 30;

this.speed += this.addSpeed\_\*(this.addSpeedNum)\*0.003;

if(this.speed > this.maxSpeed)

this.speed = this.maxSpeed;

}

else

this.addSpeedNum = 0;

}

public decSpeed(isAdd){

if(isAdd)

{

this.decSpeedNum += 1;

if(this.decSpeedNum > 30)

this.decSpeedNum = 30;

this.speed -= this.decSpeed\_\*(this.decSpeedNum)\*0.003;

if(this.speed < 0)

this.speed = 0;

}

else

this.decSpeedNum = 0;

}

public setCar(id){

this.carid = id;

var vo = this.carData[id];

this.maxSpeed = vo.maxSpeed

this.addSpeed\_ = vo.addSpeed

this.decSpeed\_ = vo.decSpeed

}

}

class GameUI extends game.BaseUI {

private static \_instance:GameUI;

public static getInstance() {

if (!this.\_instance) this.\_instance = new GameUI();

return this.\_instance;

}

private bg: eui.Image;

private errorMC: eui.Group;

private line2: eui.Image;

private line1: eui.Image;

private treeGroup: eui.Group;

private titleMC: eui.Image;

private startBtn: eui.Image;

private rankBtn: eui.Image;

private levelBtn: eui.Image;

private skinBtn: eui.Image;

private settingBtn: eui.Image;

private levelGroup: eui.Group;

private levelText: eui.Label;

private readyText: eui.Label;

private speedText: eui.Label;

private gameLevelText: eui.Label;

private needle: eui.Image;

private limitGroup: eui.Group;

private meterGroup: eui.Group;

private exitBtn: eui.Group;

private rateMC2: eui.Rect;

private failMC: eui.Rect;

private debugBtn: eui.Button;

private bottomGroup: eui.Group;

private infoBtn:UserInfoBtn

public posRateMC = new egret.Shape()

public cdRateMC = new egret.Shape()

public alarmMC = new egret.Shape()

private timer = new MyTimer(1000/GameData.Frame);

private carMC;

private alarm = 0

private lastDrawAlarm = 0

//private speedState = 0;

private treeArr1 = [];

private treeArr2 = [];

private treePool = [];

private carArr = [];

private road = {

1:{pos:-75},

2:{pos:-230},

3:{pos:230},

}

//private road = [-75,-230,230];

private lastCreateCarTime = 0;

private gameState = 0;

private carIndex = 5;

private touchID = {}

private addSpeedTimes

private decSpeedTimes

public constructor() {

super();

this.skinName = "GameUISkin";

}

public show() {

super.show();

}

public childrenCreated() {

super.childrenCreated();

this.alarmMC.x = this.posRateMC.x = this.cdRateMC.x = 152

this.alarmMC.y = this.posRateMC.y = this.cdRateMC.y = 154

this.meterGroup.addChildAt(this.alarmMC,1)

this.meterGroup.addChildAt(this.posRateMC,0)

this.meterGroup.addChildAt(this.cdRateMC,0)

this.meterGroup.addChild(this.posRateMC)

this.list.itemRenderer = InviteItem2;

this.slowBtn.addEventListener(egret.TouchEvent.TOUCH\_BEGIN, this.on,this);

this.speedBtn.addEventListener(egret.TouchEvent.TOUCH\_BEGIN, this.onSpeed,this);

this.addEventListener(egret.TouchEvent.TOUCH\_BEGIN, this.onTouchBegin,this);

this.addEventListener(egret.TouchEvent.TOUCH\_MOVE, this.onTouchMove,this);

this.addEventListener(egret.TouchEvent.TOUCH\_CANCEL, this.onTouchEnd,this);

this.addEventListener(egret.TouchEvent.TOUCH\_END, this.onTouchEnd,this);

this.addEventListener(egret.TouchEvent.TOUCH\_RELEASE\_OUTSIDE, this.onTouchEnd,this);

this.addBtnEvent(this.rankBtn, this.onRank);

this.addBtnEvent(this.levelBtn, this.onLevel);

this.addBtnEvent(this.skinBtn, this.onSkin);

this.addBtnEvent(this.settingBtn, this.onSetting);

this.addBtnEvent(this.exitBtn, this.reset);

this.addBtnEvent(this.debugBtn, this.onDebug);

this.debugBtn.visible = false;

this.addBtnEvent(this.startBtn, this.onStart);

this.timer.addEventListener(egret.TimerEvent.TIMER,this.onE,this)

this.carMC = new CarMC();

this.carMC.setCar(1)

this.addChildAt(this.carMC,this.carIndex);

this.carMC.scaleX = this.carMC.scaleY = 0.7;

this.carMC.horizontalCenter = 75

this.carMC.bottom = 350;

this.failMC.visible = false

this.infoBtn = new UserInfoBtn(this.startBtn, (res)=>{

//UserManager.getInstance().updateUserInfo(res,()=>{

this.renewInfo(res);

//});

}, this, Config.localResRoot + "wx\_btn\_info.png");

}

private renewInfo(res?){

var wx = window['wx'];

if(!wx)

return;

if(res)

{

this.infoBtn.visible = false;

UM.renewInfo(res.userInfo)

this.bottomGroup.visible = true;

return;

}

this.bottomGroup.visible = false;

this.startBtn.visible = false

wx.getSetting({

success: (res) =>{

console.log(res.authSetting)

if(res.authSetting["scope.userInfo"])//已授权

{

wx.getUserInfo({

success: (res) =>{

var userInfo = res.userInfo

UM.renewInfo(res.userInfo)

UM.head = userInfo.avatarUrl

UM.gender = userInfo.gender || 1 //性别 0：未知、1：男、2：女

this.bottomGroup.visible = true;

this.infoBtn.visible = false;

this.startBtn.visible = true

}

})

}

else

{

this.infoBtn.visible = true;

}

}

})

}

private onDebug(){

DebugUI.getInstance().show();

}

private showFailMC(){

this.failMC.visible = true

this.failMC.alpha = 0

egret.Tween.removeTweens(this.failMC)

egret.Tween.get(this.failMC).to({alpha:0.5},250).to({alpha:0},250).call(()=>{

this.failMC.visible = false

})

}

private onTouchBegin(e){

this.touchID[e.touchPointID] = {

x:e.stageX,

y:e.stageY,

}

//console.log('begin',e.touchPointID)

}

private onTouchMove(e){

//console.log('move',e.touchPointID)

this.touchID[e.touchPointID] = {

x:e.stageX,

y:e.stageY,

}

}

private onTouchEnd(e){

//console.log('end',e.touchPointID)

delete this.touchID[e.touchPointID];

}

private onRank(){

RankUI.getInstance().show();

}

private onLevel(){

LevelUI.getInstance().show();

}

private onSkin(){

SkinUI.getInstance().show();

}

private onSetting(){

SoundManager.getInstance().soundPlaying = !SoundManager.getInstance().soundPlaying

SoundManager.getInstance().bgPlaying = !SoundManager.getInstance().bgPlaying

this.settingBtn.source = SoundManager.getInstance().bgPlaying?'sound\_on\_btn\_png':'sound\_off\_btn\_png'

}

private onSlow(){

this.speedState = -1;

}

private onSpeed(){

this.speedState = 1;

}

public onStart(){

this.startLevel(CarManager.getInstance().maxLevel + 1)

}

public renewCar(){

this.carMC.setCar(CarManager.getInstance().skinid);

}

public startLevel(lv){

GameData.getInstance().setCar(CarManager.getInstance().skinid)

GameData.getInstance().onGameStart(lv)

this.touchID = {};

this.currentState = 'game'

this.alarm = 0

this.limitGroup.visible = false;

this.posRateMC.visible = false;

this.cdRateMC.visible = false;

this.errorMC.visible = false;

this.alarmMC.visible = false;

this.gameLevelText.text = 'LV.' + lv

}

public onShow(){

SoundManager.getInstance().playSound('road');

this.renewInfo();

this.timer.start()

this.reset();

this.renewCar();

this.addPanelOpenEvent(GameEvent.client.SKIN\_CHANGE,this.renewCar)

this.addPanelOpenEvent(GameEvent.client.INFO\_CHANGE,this.renew)

}

public reset(){

this.currentState = 'main'

this.gameState = 0;

this.line1.bottom = 0;

this.line2.bottom = 0;

this.bg.bottom = 0;

this.errorMC.visible = false;

this.levelText.text = '第 ' + (CarManager.getInstance().maxLevel + 1) + ' 关';

GameData.getInstance().isPlaying = false;

for(var s in this.road) {

var car = this.road[s].car;

if (car) {

CarMC.freeItem(car);

this.road[s].car = null;

}

}

}

private onE(){

//var t = egret.getTimer();

var GD = GameData.getInstance();

if(!GD.isPlaying)

{

if(this.gameState)

{

this.errorMC.visible = false;

var targetSpeed = this.gameState == 2?0:GD.baseSpeed;

if(GD.speed > targetSpeed)

{

GD.decSpeed(true);

}

this.speedText.text = Math.floor(GD.speed) + ''

this.onMoveBG(GD.speed);

this.onOtherCarMove();

}

else

this.onMoveBG(GD.baseSpeed);

return;

}

if(GD.countDown)

{

var num = 3-Math.floor((egret.getTimer() - GD.startTime)/1000)

if(num <= 0)

{

SoundManager.getInstance().playEffect('count\_down4')

GD.countDown = 0;

GD.startTime = egret.getTimer();//真正开始

this.readyText.text = '';

this.posRateMC.visible = true;

this.cdRateMC.visible = true;

this.resetRed();

}

else if(GD.countDown != num)

{

SoundManager.getInstance().playEffect('count\_down')

GD.countDown = num

this.readyText.text = '' + num;

egret.Tween.removeTweens(this.readyText)

this.readyText.scaleX = this.readyText.scaleY = 0

egret.Tween.get(this.readyText).to({scaleX:1.2,scaleY:1.2},250).to({scaleX:1,scaleY:1},250)

}

}

var isAdd = false;

//var isDec = false;

var isDec = true;

for(var s in this.touchID)

{

isAdd = true;

isDec = false;

}

GD.addSpeed(isAdd);

GD.decSpeed(isDec)

if(GD.speed < GD.baseSpeed)

GD.speed = GD.baseSpeed var speed = GD.speed;

this.onMoveBG(speed);

GD.onRunSpeed();

var rate = Math.min(1,GD.passMeter/GD.maxLen);

if(this.posRateMC.visible)

MyTool.getSector(159,180,rate\*180,0x000099,1,this.posRateMC)

//MyTool.getSector(159,162,rate\*216,0x000099,1,this.posRateMC)

//this.rateMC.width = 260 \*rate;

var cd = (GD.maxTime + GD.startTime - egret.getTimer());

if(cd < 0)

{

GD.isPlaying = false

this.gameState = 2;

ResultFailUI.getInstance().show();

return;

}

else if(rate >= 1)

{

GD.isPlaying = false

this.gameState = 1;

ResultUI.getInstance().show();

return;

}

if(this.cdRateMC.visible)

{

var cdRate = (egret.getTimer() - GD.startTime)/GD.maxTime;

MyTool.getSector(166,180,cdRate\*180,0xFCD550,1,this.cdRateMC);

//MyTool.getSector(166,162,cdRate\*216,0xFCD550,1,this.cdRateMC);

}

//var cd2 = Math.floor((cd%1000)/10)

//cd = Math.floor(cd/1000)

//this.cdText.text = DateUtil.getStringBySecond(cd).substr(-5) + '.' + ('00' + cd2).substr(-2)

this.speedText.text = Math.floor(GD.speed) + ''

var speedDec = GD.speed/GD.maxSpeed\*3

if(speedDec < 1)

speedDec = 0;

this.needle.rotation = 180 + GD.speed/160\*180 + (Math.random()-0.5)\*speedDec;

this.onOtherCarMove();

if(!GD.countDown && this.errorMC.visible)

{

var oo = GD.redArr[0];

var meter = oo.start - GD.passMeter //离红色的距离

var redLast = 1// - GD.pixToMeter(this.carMC.height\*this.carMC.scaleY);//离红色块的大小

this.errorMC.bottom = this.carMC.bottom + this.carMC.height\*this.carMC.scaleY + GD.meterToPix(meter)

if(this.alarm == 0 && meter <= GameData.AlertMeter && meter > GameData.AlertMeter/2) //发出警告

{

this.alarm = 1;

this.lastDrawAlarm = 0

SoundManager.getInstance().playEffect('limit\_before',()=>{

SoundManager.getInstance().playEffect('limit' + oo.speed)

})

}

this.limitGroup.visible = this.alarm == 1;

this.alarmMC.visible = this.alarm == 1;

if(this.alarmMC.visible)

{

var cdRate = oo.speed/160;

if(oo.speed <= GD.speed)

var draw = 1

else

var draw = 2

if(this.lastDrawAlarm != draw)

{

this.lastDrawAlarm = draw;

if(draw == 1)

MyTool.getSector(100,180,cdRate\*180,0xFF0000,1,this.alarmMC);

else

MyTool.getSector(100,180,cdRate\*180,0x00FF00,1,this.alarmMC);

}

}

if(meter <=0 && this.alarm < 2)//红块内

{

this.alarm = 2;

if(GD.speed > oo.speed)//超速

{

SoundManager.getInstance().playEffect('photo')

GD.startTime -= GameData.FailDecTime

this.showFailMC();

GD.redArr.shift();

this.resetRed();

this.limitGroup.visible = false;

this.alarmMC.visible = false;

}

}

else if(this.alarm == 2)

{

this.alarm = 3;

SoundManager.getInstance().playEffect('pass')

}

else if(this.alarm == 3)//过了红线

{

if(this.errorMC.bottom < -this.errorMC.height)

{

GD.redArr.shift();

this.resetRed();

}

}

if(this.limitGroup.visible)

{

this.rateMC2.width = 200\* Math.min((GameData.AlertMeter - meter)/GameData.AlertMeter,1)

}

}

}

private resetRed(){

this.alarm = 0;

var GD = GameData.getInstance();

var oo = GD.redArr[0];

if(!oo)

{

this.errorMC.visible = false;

return;

}

this.errorMC.visible = true;

}

private onOtherCarMove(){

var emptyPos = [];

for(var s in this.road)

{

var car:CarMC = this.road[s].car;

if(car)

{

var addSpeed = car.speed - GameData.getInstance().speed;

addSpeed \*= GameData.speedRate;

car.bottom += addSpeed;

if(car.bottom < - 300 || (car.bottom > GameManager.stage.stageHeight + 200))

{

CarMC.freeItem(car);

this.road[s].car = null;

}

}

else

{

emptyPos.push(this.road[s]);

}

}

if(this.gameState)

return;

if(emptyPos.length>0 && egret.getTimer() - this.lastCreateCarTime > 500 && Math.random() < 0.1)

{

this.lastCreateCarTime = egret.getTimer();

var oo = ArrayUtil.randomOne(emptyPos)

var car = CarMC.createItem();

car.scaleX = car.scaleY = 0.7;

car.setCar(Math.ceil(12\*Math.random()))

oo.car = car;

car.horizontalCenter = oo.pos;

car.speed = Math.floor(50 + Math.random()\*70)

this.addChildAt(car,this.carIndex);

if(oo.pos < 0)//右

{

car.scaleY \*= -1;

car.speed \*= -0.6;

car.bottom = GameManager.stage.stageHeight + 100;

}

else

{

if(car.speed < GameData.getInstance().speed)

car.bottom = GameManager.stage.stageHeight + 100;

else

car.bottom = -150;

}

}

}

private onMoveBG(speed){

speed \*= GameData.speedRate;

var des = 160;

this.line1.bottom -= speed

this.line2.bottom -= speed

this.bg.bottom -= speed

if(this.line1.bottom < -des)

this.line1.bottom += des

if(this.line2.bottom < -des)

this.line2.bottom += des

if(this.bg.bottom < -100)

this.bg.bottom += 100

for(var i=0;i<this.treeArr1.length;i++)

{

this.treeArr1[i].bottom -= speed;

if(this.treeArr1[i].bottom < -200)

{

this.freeTree(this.treeArr1[i])

this.treeArr1.splice(i,1);

i --;

}

}

for(var i=0;i<this.treeArr2.length;i++)

{

this.treeArr2[i].bottom -= speed;

if(this.treeArr2[i].bottom < -200)

{

this.freeTree(this.treeArr2[i])

this.treeArr2.splice(i,1);

i --;

}

}

var treeDes = 150;

while(!this.treeArr1[0] || this.treeArr1[this.treeArr1.length-1].bottom < GameManager.stage.stageHeight-treeDes){

var tree = this.getTree();

if(!this.treeArr1[0])

tree.bottom = 0;

else

tree.bottom = this.treeArr1[this.treeArr1.length-1].bottom + treeDes + Math.random()\*20;

this.treeArr1.push(tree);

this.treeGroup.addChildAt(tree,0);

tree.right = 590

}

while(!this.treeArr2[0] || this.treeArr2[this.treeArr2.length-1].bottom < GameManager.stage.stageHeight-treeDes){

var tree = this.getTree();

if(!this.treeArr2[0])

tree.bottom = 0;

else

tree.bottom = this.treeArr2[this.treeArr2.length-1].bottom + treeDes + Math.random()\*20//treeDes;

this.treeArr2.push(tree);

this.treeGroup.addChildAt(tree,0);

//this.addChild(this.speedBtn);

tree.right = -30

}

}

private getTree(){

var mc:eui.Image = this.treePool.pop();

if(!mc)

{

mc = new eui.Image()

mc.source = 'tree\_'+Math.ceil(2+Math.random()\*6)+'\_png'

}

return mc;

}

private freeTree(mc){

MyTool.removeMC(mc);

this.treePool.push(mc);

}

}

class LevelItem extends game.BaseItem{

private indexText: eui.Label;

private s0: eui.Image;

private s1: eui.Image;

private s2: eui.Image;

public constructor() {

super();

this.skinName = "LevelItemSkin";

}

public childrenCreated() {

this.addBtnEvent(this,this.onClick)

}

private onClick(){

if(this.currentState == 'current')

{

LevelUI.getInstance().hide();

GameUI.getInstance().startLevel(this.data)

}

}

public dataChanged():void{

this.indexText.text = this.data;

var star = CarManager.getInstance().getLevelStar(this.data)

if(star || this.data == CarManager.getInstance().maxLevel + 1)

{

this.currentState = 'current'

for(var i=0;i<3;i++)

{

this['s' + i].source = i<star?'chapter\_star4\_png':'chapter\_star3\_png'

}

}

else

{

this.currentState = 'lock'

}

}

}

class LevelUI extends game.BaseWindow{

private static \_instance:LevelUI;

public static getInstance() {

if (!this.\_instance) this.\_instance = new LevelUI();

return this.\_instance;

}

private closeBtn: eui.Group;

private scroller: eui.Scroller;

private list: eui.List;

public constructor() {

super();

this.skinName = "LevelUISkin";

}

public childrenCreated() {

this.addBtnEvent(this.closeBtn,this.hide)

this.scroller.viewport = this.list;

this.list.itemRenderer = LevelItem;

}

public show() {

super.show();

}

public onShow(){

var arr = [];

for(var i=0;i<GameData.MaxLevel;i++)

arr.push(i+1)

this.list.dataProvider = new eui.ArrayCollection(arr)

}

}

class RankUI extends game.BaseWindow{

private static \_instance:RankUI;

public static getInstance() {

if (!this.\_instance) this.\_instance = new RankUI();

return this.\_instance;

}

private closeBtn: eui.Group;

private bitmap: egret.Bitmap;

private isdisplay = false;

private isSendConfig:boolean;

private isLoadFile:boolean;

public constructor() {

super();

this.skinName = "RankUISkin";

}

public childrenCreated() {

super.childrenCreated();

this.addBtnEvent(this.closeBtn,this.hide)

this.touchEnabled = false;

}

public onShow(){

if(!window['wx'])

return;

this.showBitmapList();

}

private poseData(){

let param:any = {

me: UM.gameid,

command: 'open',

x:this.bitmap.x + (GameManager.uiWidth - this.width)/2,

y:this.bitmap.y + (GameManager.uiHeight - this.height)/2,

me\_value: CarManager.getInstance().getAllStar() + ',0', //第2位时间传0，永远排在最上面

root: "openDataContext/",

}

//发送消息

var platform = window['platform']

platform.openDataContext.postMessage(param);

}

//0 好友榜，2群排行

public showBitmapList(){

if(!window["wx"] || !window["wx"].getOpenDataContext) return;

this.remove();

var platform = window['platform']

if (!this.isdisplay) {

this.bitmap = platform.openDataContext.createDisplayObject(null, this.stage.stageWidth, this.stage.stageHeight);

this.bitmap.x = 20;

this.bitmap.y = 110;

this.addChild(this.bitmap);

this.bitmap.touchEnabled = false

this.isdisplay = true;

this.poseData();

}

}

protected remove(){

var platform = window['platform']

if(this.isdisplay){

this.isdisplay = false;

this.bitmap.parent && this.bitmap.parent.removeChild(this.bitmap);

if(platform.openDataContext){

platform.openDataContext.postMessage({ command: 'close' });

}

}

}

public hide(){

this.remove();

super.hide();

}

}

class ResultFailUI extends game.BaseWindow{

private static \_instance:ResultFailUI;

public static getInstance() {

if (!this.\_instance) this.\_instance = new ResultFailUI();

return this.\_instance;

}

private rateText: eui.Label;

private timesText: eui.Label;

private retryBtn: eui.Button;

private backBtn: eui.Button;

public constructor() {

super();

this.skinName = "ResultFailUISkin";

this.canBGClose = false

}

public childrenCreated() {

this.addBtnEvent(this.backBtn,this.hide)

this.addBtnEvent(this.retryBtn,this.onRetry)

}

private onRetry(){

this.hide();

GameUI.getInstance().startLevel(GameData.getInstance().level);

}

public hide(){

super.hide();

GameUI.getInstance().reset();

}

public show() {

super.show();

}

public onShow(){

SoundManager.getInstance().playEffect('die')

var GD = GameData.getInstance();

var rate = Math.min(1,GD.passMeter/GD.maxLen);

this.rateText.text = Math.floor(rate\*100)+'%'

if(CarManager.getInstance().maxLevel < GD.level)

{

var failTime = SharedObjectManager.getInstance().getMyValue('chapter'+GD.level) || 0;

failTime ++;

SharedObjectManager.getInstance().setMyValue('chapter'+GD.level,failTime);

this.timesText.text = '已尝试'+failTime+'次'

}

else

this.timesText.text = '';

}

class ResultUI extends game.BaseWindow{

private static \_instance:ResultUI;

public static getInstance() {

if (!this.\_instance) this.\_instance = new ResultUI();

return this.\_instance;

}

private timeText: eui.Label;

private nextText: eui.Label;

private shartBtn: eui.Button;

private nextBtn: eui.Button;

private backBtn: eui.Group;

private useTime;

public constructor() {

super();

this.skinName = "ResultUISkin";

this.canBGClose = false

}

public childrenCreated() {

this.addBtnEvent(this.backBtn,this.hide)

this.addBtnEvent(this.nextBtn,this.onNext)

this.addBtnEvent(this.shartBtn,this.onShare)

this.shartBtn.visible = false;

}

private onShare(){

}

private onNext(){

this.hide();

GameUI.getInstance().startLevel(GameData.getInstance().level + 1);

}

public hide(){

super.hide();

GameUI.getInstance().reset();

}

public show() {

this.useTime = egret.getTimer() - GameData.getInstance().startTime;

CarManager.getInstance().sendSuccess(this.useTime,()=>{

super.show();

})

}

public onShow(){

SoundManager.getInstance().playEffect('win')

var GD = GameData.getInstance();

var star = GD.getStarByLevel(GD.level,this.useTime)

for(var j=0;j<3;j++)

{

this['s'+j].source = star>j?'chapter\_star2\_png':'chapter\_star1\_png'

}

this.nextText.visible = GD.level == CarManager.getInstance().maxLevel;

this.nextBtn.visible = GD.level != GameData.MaxLevel

var cd = this.useTime;

var cd2 = Math.floor((cd%1000)/10)

cd = Math.floor(cd/1000)

this.timeText.text = DateUtil.getStringBySecond(cd).substr(-5) + '.' + ('00' + cd2).substr(-2)

}

}

class SkinItem extends game.BaseItem{

private bg: eui.Image;

private desText: eui.Label;

private rateText: eui.Label;

private barMC: eui.Image;

private btn: eui.Button;

private carMC = new CarMC();

public constructor() {

super();

this.skinName = "SkinItemSkin";

}

public childrenCreated() {

this.addBtnEvent(this,this.onClick)

this.addBtnEvent(this.btn,this.onBuy)

this.addChildAt(this.carMC,1);

this.carMC.horizontalCenter = 0;

this.carMC.y = 8;

this.carMC.scaleX = this.carMC.scaleY = 0.5;

}

private onBuy(e){

e.stopImmediatePropagation()

if(this.btn.label == '解锁')

{

CarManager.getInstance().buySkin(this.data,()=>{

});

return;

}

var carVO = GameData.getInstance().carData[this.data];

switch(carVO.buyType)

{

case 0://默认

break

case 1:

case 5:

SkinUI.getInstance().hide();

GameUI.getInstance().onStart()

break;

case 2:

break;

case 3:

console.log(UM.gameid);

ShareTool.share('集赞换新车，就差你了一票了',Config.localResRoot + "share\_img\_1.jpg",{type:1,from:UM.gameid,skinid:this.data})

break;

case 6:

CarManager.getInstance().shareNum ++;

ShareTool.share('这个游戏很好玩，推荐一下',Config.localResRoot + "share\_img\_2.jpg",{})

this.dataChanged();

break;

case 4:

break;

}

}

private onClick(){

if(SkinUI.getInstance().currentSkin == this.data)

return;

if(!CarManager.getInstance().isHaveSkin(this.data))

return;

SkinUI.getInstance().currentSkin = this.data;

EM.dispatch(GameEvent.client.SKIN\_CHANGE)

SoundManager.getInstance().playEffect('switch')

}

private onTimer(){

if(this.currentState == 'lock' && GameData.getInstance().carData[this.data].buyType == 2)

this.dataChanged();

}

public dataChanged():void{

var openNum = CarManager.getInstance().skinNum;

if(this.data > openNum + 1)

{

this.currentState = 'lock2';

this.carMC.visible = false

this.cacheAsBitmap = false;

return;

}

var carVO = GameData.getInstance().carData[this.data];

this.carMC.visible = true

this.carMC.setCar(this.data)

if(CarManager.getInstance().isHaveSkin(this.data))

{

var color = 0x99c5fd;

this.setUsing(SkinUI.getInstance().currentSkin == this.data);

this.currentState = 'normal';

this.setHtml(this.desText, this.createHtml('时速：',color) + carVO.maxSpeed+'KM\n'+

this.createHtml('加速：',color) + carVO.addSpeed+'KM\n'+

this.createHtml('制动：',color) + carVO.decSpeed+'KM');

}

else

{

this.cacheAsBitmap = false;

this.currentState = 'lock';

var v = CarManager.getInstance().getSkinValue(this.data)

var max = carVO.buyValue;

this.btn.visible = true;

switch(carVO.buyType)

{

case 0://默认

break

case 1:

this.rateText.text = '完成第'+max+'关'

this.btn.label = '前往'

break

case 2:

if(max < 1)

this.rateText.text = Math.round(max\*60) + '分钟后获得'

else

this.rateText.text = max + '小时后获得'

this.btn.visible = false;

break

case 3:

this.rateText.text = '邀请'+max+'个新玩家'

this.btn.label = '邀请'

break

case 4:

this.rateText.text = '观看广告'+max+'次'

this.btn.label = '观看'

break

case 5:

this.rateText.text = '总星星数'+max+'个'

this.btn.label = '前往'

break

case 6:

this.rateText.text = '推荐游戏'

this.btn.label = '推荐'

break

}

this.barMC.width = 240\*Math.min(v/max,1)

if(v>=max)

{

this.btn.label = '解锁'

this.btn.visible = true;

}

}

}

public setUsing(b){

//return;

if(!b)

{

this.carMC.filters = [];

this.cacheAsBitmap = false;

return;

}

var color:number = 0x33CCFF;

var alpha:number = 0.8;

var blurX:number = 80;

var blurY:number = 80;

var strength:number = 2;

var quality:number = egret.BitmapFilterQuality.HIGH;

var inner:boolean = false;

var knockout:boolean = false;

var glowFilter:egret.GlowFilter = new egret.GlowFilter( color, alpha, blurX, blurY,

strength, quality, inner, knockout );

this.carMC.filters = [glowFilter];

}

}

class SkinUI extends game.BaseWindow{

private static \_instance:SkinUI;

public static getInstance() {

if (!this.\_instance) this.\_instance = new SkinUI();

return this.\_instance;

}

private closeBtn: eui.Group;

private scroller: eui.Scroller;

private list: eui.List;

public currentSkin

public constructor() {

super();

this.skinName = "SkinUISkin";

}

public childrenCreated() {

this.addBtnEvent(this.closeBtn,this.hide)

this.scroller.viewport = this.list;

this.list.itemRenderer = SkinItem;

}

public show() {

super.show();

}

public onShow(){

this.currentSkin = CarManager.getInstance().skinid;

this.addPanelOpenEvent(GameEvent.client.SKIN\_CHANGE,this.renewList)

var arr = [];

for(var i=0;i<GameData.MaxCar;i++)

arr.push(i+1)

this.list.dataProvider = new eui.ArrayCollection(arr)

this.validateNow();

var max = this.scroller.viewport.contentHeight-this.scroller.height;

var v = Math.min(max,Math.floor(CarManager.getInstance().skinNum/3)\*(260+15));

egret.Tween.get(this.scroller.viewport).to({scrollV:v},100)

}

private renewList(){

MyTool.renewList(this.list)

}

private onTimer(){

MyTool.runListFun(this.list,'onTimer')

}

public hide(){

super.hide();

if(this.currentSkin != CarManager.getInstance().skinid)

{

CarManager.getInstance().setCarSkin(this.currentSkin,()=>{

});

}

}

}

}

class EventManager extends egret.EventDispatcher {

private static \_instance: EventManager;

public static getInstance():EventManager{

if(!EventManager.\_instance)

class Net extends egret.EventDispatcher{

private static instance: Net;

public static getInstance() {

if(!this.instance) this.instance = new Net();

return this.instance;

}

//发送消息

public static send(head,msg,fun?,isMode = true,serverType=1,isRepeat=false){

Net.getInstance().send(head,msg,fun,isMode,serverType,isRepeat);

}

//添加用户信息

public static addUser(msg){

//msg.landid = UM.landid;

msg.gameid = UM.gameid;

}

private msgIndex = 0;

public modeNum = 0;

public outPut = true;

public actionRecord = []

public constructor() {

super();

this.modeNum = 0;

}

private getVariables(head,msg,serverType)

{

var variables = new egret.URLVariables('a=1');

var oo:any = {};

oo.head = head;

if(msg.gameid)

{

oo.msg\_index = this.msgIndex;

this.msgIndex ++;

}

oo.msg = JSON.stringify(msg);

oo.debug\_client = Config.isDebug;

if(serverType == 1)

oo.version = Config.version;

else

oo.version = Config.user\_version;

if(\_get['debug\_server'])

oo.debug\_server = 1;

if(\_get['game\_version'])

oo.game\_version = \_get['game\_version'];

return variables;

}

private pushRecord(str){

this.actionRecord.push(str)

while(this.actionRecord.length > 10)

this.actionRecord.shift();

}

public send(head,msg,fun?,isMode = true,serverType=1,isRepeat=false){

var loader = new egret.URLLoader();

loader.dataFormat = egret.URLLoaderDataFormat.TEXT;

loader['isMode'] = isMode

loader['fun'] = fun

loader['head'] = head

loader['msg'] = msg;

loader['isRepeat'] = isRepeat;

if(serverType == 1)

{

var request = new egret.URLRequest(Config.host);

}

else

var request = new egret.URLRequest(Config.userHost);

//var request = new egret.URLRequest('http://qxu1606510485.my3w.com/new\_index.php');

request.method = egret.URLRequestMethod.POST;

request.data = this.getVariables(head,msg,serverType);

if(Config.isDebug && this.outPut)

{

console.log('send===> '+head)

console.log(JSON.stringify(msg) +' '+TM.now());

}

if(isMode)

{

this.modeNum ++;

//GameManager.container.touchChildren = GameManager.container.touchEnabled = false;

this.addLoading();

}

this.pushRecord('send:' + head + '#' + this.modeNum)

loader.load(request);

loader.addEventListener(egret.Event.COMPLETE, this.onComplete, this);

loader.addEventListener(egret.IOErrorEvent.IO\_ERROR, this.onError, this);

}

private onComplete(e){

var loader = e.currentTarget;

var isMode = loader.isMode

var fun = loader.fun

var head = loader.head

var msg = loader.msg

if(isMode)

this.modeNum --;

if(this.modeNum <= 0)

{

//GameManager.container.touchChildren = GameManager.container.touchEnabled = true;

this.removeLoading();

}

this.pushRecord('receive:' + head + '#' + this.modeNum)

loader.removeEventListener(egret.Event.COMPLETE, this.onComplete, this);

loader.removeEventListener(egret.IOErrorEvent.IO\_ERROR, this.onError, this);

if(Config.isDebug && this.outPut)

{

console.log('====receive> '+head)

console.log(e.target.data +' ' +TM.now());

}

try {

var data = JSON.parse(e.target.data)

if(Config.isDebug && this.outPut)

console.log(data.msg);

}catch(e){

if(Config.isDebug)

MyWindow.Alert('通信数据异常');

else

MyWindow.Alert('通信数据异常',this.refresh,'重新登陆');

return;

}

if(data.error)

{

this.removeLoading();

switch (data.error)

{

case 1:

if(\_get['app'])

MyWindow.Alert('游戏已更新，请重新下载');

else

MyWindow.Alert('游戏已更新，请退出重进',this.refresh,'重新登陆');

GameManager.getInstance().stopTimer();

break;

case 2:

MyWindow.Alert('该用户已在其它地方登录',this.refresh,'重新登陆');

GameManager.getInstance().stopTimer();

break;

case 3:

MyWindow.Alert('通信出错',this.refresh,'重新登陆');

break;

case 4:

MyWindow.Alert('用户数据写入失败',this.refresh,'重新登陆');

GameManager.getInstance().stopTimer();

break;

case 5:

MyWindow.Alert('服务器正在维护中，请稍后再试',this.refresh);

GameManager.getInstance().stopTimer();

break;

case 99:

MyWindow.Alert(data.error\_str,this.refresh);

GameManager.getInstance().stopTimer();

break;

default :

MyWindow.Alert('未知错误',this.refresh);

GameManager.getInstance().stopTimer();

break;

}

return;

}

TM.init(data.server\_time);

//SyncDataManager.getInstance().snyc(data.msg);

var oo:any = {};

oo.sendData = msg;

oo.msg = data.msg;

if(fun)

fun(oo);

this.dispatchEventWith(head,false,oo);

}

private onError(e){

var loader = e.currentTarget;

loader.removeEventListener(egret.Event.COMPLETE, this.onComplete, this);

loader.removeEventListener(egret.IOErrorEvent.IO\_ERROR, this.onError, this);

if(loader.isRepeat)

{

MyWindow.Confirm('数据提交失败，与服务器失去连接！',(b)=>{

if(b==1)

{

this.send(loader.head,loader.msg,loader.fun,loader.isMode,1,true);

}

else

this.refresh();

},['刷新', '重试']);

}

else

{

MyWindow.Alert('与服务器失去连接！',this.refresh);

GameManager.getInstance().stopTimer();

}

this.pushRecord('error' + this.modeNum)

this.removeLoading();

}

private refresh(){

location.reload();

}

private addLoading(){

MsgingUI.getInstance().show();

}

private removeLoading(){

MsgingUI.getInstance().hide();

}

}

class MyWindow {

public constructor() {

}

public static Alert(msg, fun?, btnLabel?) {

if (\_get['debug'] == 100) {

document.body.style.background = '#000000'

return;

}

var panel = new AlertUI();

panel.show(msg, fun, btnLabel);

}

public static AlertRelogin(msg) {

if (\_get['debug'] == 100) {

document.body.style.background = '#000000'

return;

}

var panel = new AlertUI();

panel.show(msg, ()=>{

MyTool.refresh();

});

GameManager.getInstance().stopTimer();

}

//fun(type){type:1确定，2：取消，3右上角关闭}

public static Confirm(msg, fun?, btnWord = ['取消', '确定'], sp?) {

var panel = new ConfirmUI();

panel.show(msg, fun, btnWord, sp)

return panel

}

public static ShowTips(msg, cd = 0) {

if(!cd)

cd = Math.max(1000,msg.length \* 150)

TipsUI.getInstance().show(msg, cd);

}

public static addBtnTips(mc, str, addStageRemove?) {

var timer;

mc.touchEnabled = true;

mc.addEventListener(egret.TouchEvent.TOUCH\_BEGIN, onTouchStart, mc.thisObj || mc);

if (addStageRemove) {

GameManager.stage.once(egret.Event.REMOVED\_FROM\_STAGE, function () {

mc.removeEventListener(egret.TouchEvent.TOUCH\_BEGIN, onTouchStart, mc.thisObj || mc);

}, this)

}

function onTouchStart(e)

{

var stageX = e.stageX

var stageY = e.stageY

mc.stage.removeEventListener(egret.TouchEvent.TOUCH\_END, onTouchEnd, mc.thisObj || mc);

mc.stage.once(egret.TouchEvent.TOUCH\_END, onTouchEnd, mc.thisObj || mc);

mc.stage.once(egret.TouchEvent.TOUCH\_CANCEL, onTouchEnd, mc.thisObj || mc);

egret.clearTimeout(timer);

timer = egret.setTimeout(function () {

if (Math.abs(GameManager.stageX - stageX) > 20 || Math.abs(GameManager.stageY - stageY) > 20) {

mc.stage.removeEventListener(egret.TouchEvent.TOUCH\_END, onTouchEnd, mc.thisObj || mc);

mc.stage.removeEventListener(egret.TouchEvent.TOUCH\_CANCEL, onTouchEnd, mc.thisObj || mc);

return;

}

if (typeof str == 'string')

TouchTipsUI.getInstance().show(e, str);

else

TouchTipsUI.getInstance().show(e, str.apply(mc.thisObj || mc));

}, this, 300)

}

function onTouchEnd()

{

mc.stage.removeEventListener(egret.TouchEvent.TOUCH\_END, onTouchEnd, mc.thisObj || mc);

mc.stage.removeEventListener(egret.TouchEvent.TOUCH\_CANCEL, onTouchEnd, mc.thisObj || mc);

if (TouchTipsUI.getInstance().stage)

mc.stopClickTimer = egret.getTimer();

egret.clearTimeout(timer);

TouchTipsUI.getInstance().hide();

}

}

}

class SharedObjectManager {

public constructor() {

}

private static \_instance: SharedObjectManager;

public static getInstance():SharedObjectManager{

if(!SharedObjectManager.\_instance)

SharedObjectManager.\_instance = new SharedObjectManager();

return SharedObjectManager.\_instance;

}

private getUserSign():string{

return UM.gameid;

}

public setMyValue(key:string,value:any){

key = this.getUserSign() + "\_" + key; this.setValue(key,value);

}

public getMyValue(key:string):any{

key = this.getUserSign() + "\_" + key;

return this.getValue(key);

}

public removeMyValue(key:string) {

key = this.getUserSign() + "\_" + key;

delete localStorage[key];

}

public setValue(key:string,value:any) {

var data:any = {};

data.data = value;

data = JSON.stringify(data);

localStorage[key] = data;

}

public getValue(key:string):any {

var value = localStorage[key];

if(!value)

return null;

var data = JSON.parse(value);

return data.data;

}

}

class SoundManager {

private static instance: SoundManager;

public constructor() {

this.init();

}

public static getInstance(): SoundManager {

if(!this.instance)

this.instance = new SoundManager();

return this.instance;

}

//默认关闭音乐

private \_soundPlaying: boolean = false;

private \_bgPlaying: boolean = false;

private \_openShake: boolean = true;

private \_isPlayMovie: boolean = true;

private \_isMessage: boolean = true;

private currentChannel:egret.SoundChannel;

private currentKey :string;

private bgKey :string;

private lastBGKey :string;

private isLoad:boolean=false;

private bgTimer;

public pkKey = [];

public effectKey = [];

// private tween:egret.Tween

private init(){

this.\_soundPlaying = true;

this.\_bgPlaying = true;

var som = SharedObjectManager.getInstance();

if(som.getValue("sound") != undefined)

this.\_soundPlaying = som.getValue("sound");

if(som.getValue("bgsound") != undefined)

this.\_bgPlaying = som.getValue("bgsound");

if(som.getValue("openShake") != undefined)

this.\_openShake = som.getValue("openShake");

if(som.getValue("playMovie") != undefined)

this.\_isPlayMovie = som.getValue("playMovie");

if(som.getValue("showMessage") != undefined)

this.\_isMessage = som.getValue("showMessage");

this.isLoad=this.\_soundPlaying;

}

public get soundPlaying(){

// if(!Config.isDebug && !Config.testSound) return false;

return this.\_soundPlaying

}

public get bgPlaying(){

// if(!Config.isDebug && !Config.testSound) return false;

return this.\_bgPlaying

}

public get openShake(){

return this.\_openShake

}

public get isPlayMovie(){

return this.\_isPlayMovie

}

public get isMessage(){

return this.\_isMessage

}

public set soundPlaying(v){

if(this.\_soundPlaying!=v)

SharedObjectManager.getInstance().setValue("sound",v)

this.\_soundPlaying = v;

//this.loadEffectSound();

}

public set bgPlaying(v){

if(this.\_bgPlaying!=v){

SharedObjectManager.getInstance().setValue("bgsound",v);

}

this.\_bgPlaying= v;

if(!v ){

this.stopBgSound();

}

else{

this.playSound('road');

}

}

public set openShake(v){

if(this.\_openShake!=v)

SharedObjectManager.getInstance().setValue("openShake",v)

this.\_openShake= v;

}

public set isPlayMovie(v){

if(this.\_isPlayMovie!=v)

SharedObjectManager.getInstance().setValue("playMovie",v)

this.\_isPlayMovie= v;

}

public set isMessage(v){

if(this.\_isMessage!=v)

SharedObjectManager.getInstance().setValue("showMessage",v)

this.\_isMessage= v;

}

public addBtnClickEffect(btn:egret.DisplayObject){

btn.addEventListener(egret.TouchEvent.TOUCH\_TAP,function(){

this.playBtn();

},this)

}

public playBtn(){

this.playEffect('btn');

}

public stopBgSound(){

this.lastBGKey = this.bgKey;

this.bgKey = null;

try{

egret.clearTimeout(this.playTime);

if(this.currentChannel){

egret.Tween.removeTweens(this.currentChannel);

this.currentChannel.stop();

}

this.onSoundComplete();

}catch(e){}

}

public playEffect(v:string, fun?,thisObj?){

if(!this.soundPlaying) return;

console.log('call:',v)

var url = "resource/sound/" + v +".mp3"

var loader: egret.URLLoader = new egret.URLLoader();

loader.dataFormat = egret.URLLoaderDataFormat.SOUND;

loader.once(egret.Event.COMPLETE,()=>{

var sound: egret.Sound = <egret.Sound>loader.data;

var channel = sound.play(0,1);

console.log(v)

if(fun)

channel.once(egret.Event.SOUND\_COMPLETE,fun,thisObj)

},this);

loader.load(new egret.URLRequest(url));

}

public resumeSound(){

if(this.lastBGKey)

this.playSound(this.lastBGKey);

}

private tempLoop:number;

public playSound(key:string, loop:number = 9999){

//console.log(key)

if(!this.bgPlaying) return;

if(this.bgKey == key) return;

this.bgKey = key;

var url = "resource/sound/" + key +".mp3"

if(this.currentKey == url) return;

this.currentKey=url;

try{

this.tempLoop = loop;

/\*if(this.currentChannel && this.currentKey == url){

return;

}

this.currentKey=url\*/

var loader: egret.URLLoader = new egret.URLLoader();

loader.dataFormat = egret.URLLoaderDataFormat.SOUND;

loader.addEventListener(egret.Event.COMPLETE,this.onLoadComplete,this);

loader.addEventListener(egret.IOErrorEvent.IO\_ERROR,this.onLoadError,this);

loader.load(new egret.URLRequest(url));

}

catch(e){

}

}

private playTime:number;

private onLoadComplete(event: egret.Event): void {

egret.clearTimeout(this.playTime);

var loader: egret.URLLoader = <egret.URLLoader>event.target;

var self = this;

try{

this.onLoadError(event);

if((this.currentKey && loader.data.url != this.currentKey) || !this.\_bgPlaying)

return;

if(this.currentChannel){

self.currentChannel.stop();

self.currentChannel=null;

if(!self.\_bgPlaying)return; }

else

fun();

}

catch(e){

}

function fun(){

var sound: egret.Sound = <egret.Sound>loader.data;

var channel: egret.SoundChannel = sound.play(0,self.tempLoop);

self.currentChannel = channel;

channel.addEventListener(egret.Event.SOUND\_COMPLETE, self.onSoundComplete, self);

}

}

private onSoundComplete(event?:egret.Event):void {

this.currentChannel = null;

this.currentKey = null;

}

private onLoadError(event: egret.Event): void {

var loader: egret.URLLoader = <egret.URLLoader>event.target;

loader.removeEventListener(egret.Event.COMPLETE,this.onLoadComplete,this);

loader.removeEventListener(egret.IOErrorEvent.IO\_ERROR,this.onLoadError,this);

}

class TimeManager {

public constructor() {

}

private static \_instance: TimeManager;

public static getInstance():TimeManager{

if(!TimeManager.\_instance)

TimeManager.\_instance = new TimeManager();

return TimeManager.\_instance;

}

public timeDiff: number = 0;

public init(time:number):void{

//本地和服务器的时间差

this.timeDiff = Math.floor(Date.now() / 1000 - time);

}

public now():number{

return Math.floor(Date.now() / 1000) - this.timeDiff //+ 24\*3600 \*7;

}

public nowMS():number{

return Date.now() - this.timeDiff\*1000

}

public getLastDayOfWeekDate(time:number, endDay:any):Date{

endDay = endDay || 5;

//得到今天是周几

var d = new Date(time \* 1000);

var curDay = d.getDay();

var add = endDay > curDay

? endDay - curDay

: 7 - (curDay - endDay);

return new Date(d.getTime() + add \* 24 \* 3600 \* 1000);

}

public offsetDate():Date{

var offsetTime = -21600;

var time = this.now();

time += offsetTime;

return DateUtil.timeToChineseDate(time);

}

public chineseDate():Date{

return DateUtil.timeToChineseDate(this.now());

}

public getNextDateTime():number{

return DateUtil.getNextDateTimeByHours(6);

}

}

class UserManager {

public constructor() {

}

private static \_instance: UserManager;

public static getInstance():UserManager{

if(!UserManager.\_instance)

UserManager.\_instance = new UserManager();

return UserManager.\_instance;

}

public gameid: string;

public nick: string;

public head: string;

public dbid: string;

public gender: number;

public isScope: boolean = false;

public fill(data:any):void{

this.dbid = data.\_id;

CarManager.getInstance().initData(data);

}

public renewInfo(userInfo){

this.isScope = true;

this.nick = userInfo.nickName

this.head = userInfo.avatarUrl

this.gender = userInfo.gender || 1 //性别 0：未知、1：男、2：女

}

public getUserInfo(fun){

var wx = window['wx'];

const db = wx.cloud.database();

wx.cloud.callFunction({ //取玩家openID,

name: 'getInfo',

complete: (res) => {

console.log(res)

this.gameid = res.result.openid

db.collection('user').where({ //取玩家数据

\_openid: this.gameid,

}).get({

success: (res)=>{

if(res.data.length == 0)//新用户

{

this.onNewUser(fun)

return;

}

//this.testAddInvite()//debug

this.fill(res.data[0]);

fun && fun();

}

})

}

})

}

private testAddInvite(){

var wx = window['wx'];

var query = wx.getLaunchOptionsSync().query;

if(query.type == '1')

{

wx.cloud.callFunction({ //取玩家openID,

name: 'onShareIn',

data:{

other:query.from,

skinid:query.skinid,

},

complete: (res) => {

//console.log(res)

}

})

}

}

//新用户注册

private onNewUser(fun?){

var wx = window['wx'];

const db = wx.cloud.database();

var initData:any = {

skins:[1], //已有皮肤

skinid:1, //当前使用皮肤

skinsData:{}, //皮肤相关的额外数据

levelData:{}, //关卡数据

};

db.collection('user').add({

data:initData,

success: (res)=>{

initData.\_id = res.\_id;

this.fill(initData);

fun && fun();

}

})

this.testAddInvite();

}

}

//界面显示 Banner广告

class DebugWXGameAD {

private con:eui.Rect = new eui.Rect();

public createBannerAd(btnx,btny,btnw){

this.con.x = btnx;

this.con.y = btny;

this.con.width = btnw;

this.con.height = btnw \* 0.35;

}

public hide(){

MyTool.removeMC(this.con);

}

public show():void{

GameManager.stage.addChild(this.con);

}

public destroy(){

this.hide();

}

public offLoad(){}

public offError(){}

public setY(btny:number){

this.con.y = btny;

//WXAddCode.execute();

}

}

class AlertUI extends game.BaseWindow {

public constructor() {

super();

this.skinName = "AlertSkin";

}

private cancelBtn: eui.Button;

private okBtn: eui.Button;

private closeBtn: eui.Button;

private text: eui.Label;

private textIn;

private fun;

private btnLabel;

public childrenCreated() {

this.canBGClose = false;

super.childrenCreated();

this.addBtnEvent(this.okBtn, this.onClick);

MyTool.removeMC(this.closeBtn)

MyTool.removeMC(this.cancelBtn)

}

public show(v?,fun?,btnLabel?){

this.textIn = v;

this.fun = fun;

this.btnLabel = btnLabel;

super.show();

}

public onShow(){

MyTool.setColorText(this.text, this.textIn);

this.okBtn.label = this.btnLabel || '确认'

if(this.text.numLines > 1)

this.text.textAlign = 'left'

var ww = GameManager.container.width;

var hh = GameManager.container.height;

this.x = (ww - this.width) / 2;

this.y = (hh - this.height) / 2;

}

private onClick(){

this.hide();

if(this.fun)

this.fun();

}

}

class BottomUI extends game.BaseContainer {

public constructor() {

super();

}

public closeBtn: eui.Button;

private hideFun;

private thisObj;

public childrenCreated() {

this.addBtnEvent(this.closeBtn,this.backBtnClick);

}

public setHide(fun,thisObj):void{

this.hideFun = fun;

this.thisObj = thisObj;

}

private backBtnClick(event:egret.TouchEvent):void {

this.hideFun && this.hideFun.apply(this.thisObj);

}

}

class CommonNumInput extends game.BaseItem{

public static RENEW:string = "renew";

public constructor() {

super();

this.skinName = "CommonNumInputSkin";

}

private reduceBtn: eui.Image;

private addBtn: eui.Image;

private input: eui.EditableText;

private \_maxNum:number = 999;

public minNum:number=1;

public stepNum:number = 1;//点击一次增加的数值;

public addStr = ''

public childrenCreated() {

super.childrenCreated();

this.addBtn.addEventListener(egret.TouchEvent.TOUCH\_BEGIN,this.onStart,this);

this.addBtn.addEventListener(egret.TouchEvent.TOUCH\_END,this.onEnd,this);

this.addBtn.addEventListener(egret.TouchEvent.TOUCH\_RELEASE\_OUTSIDE,this.onEnd,this)

this.reduceBtn.addEventListener(egret.TouchEvent.TOUCH\_RELEASE\_OUTSIDE,this.onEnd,this)

this.reduceBtn.addEventListener(egret.TouchEvent.TOUCH\_BEGIN,this.onStart,this);

this.reduceBtn.addEventListener(egret.TouchEvent.TOUCH\_END,this.onEnd,this);

this.input.addEventListener(egret.TextEvent.FOCUS\_OUT,this.onOut,this);

this.input.addEventListener(egret.TextEvent.FOCUS\_IN,this.onIn,this);

this.input.addEventListener(egret.TextEvent.CHANGE,this.onChange,this);

}

public set maxNum(v){

this.\_maxNum = v;

this.input.maxChars = String(v).length+1

}

public get maxNum(){

return this.\_maxNum;

}

private startTime:number

private timer:any;

private addNum:number

private \_nowNum:number;

private onStart(e:egret.Event) {

e.preventDefault();

this.startTime = TM.now();

this.timer = egret.setInterval(this.onTimer,this,500);

this.addNum = (e.currentTarget == this.addBtn ? 1 : -1) \* this.stepNum;

MyTool.changeGray(this.addBtn,false,true);

MyTool.changeGray(this.reduceBtn,false,true);

this.onTimer(true);

}

private onTimer(bool) {

this.\_nowNum += this.addNum;

if(this.\_nowNum >= this.\_maxNum) {

this.\_nowNum = this.\_maxNum;

this.addNum = 0;

MyTool.changeGray(this.addBtn,true,true);

}

else if(this.\_nowNum <= this.minNum ) {

this.\_nowNum =this.minNum;

this.addNum = 0;

MyTool.changeGray(this.reduceBtn,true,true);

}

if(!bool && this.addNum != 0)

this.timer = egret.setInterval(this.onTimer,this,30);

this.input.text = this.\_nowNum + this.addStr;

this.renewCoin();

}

private onEnd(e) {

egret.clearInterval(this.timer);

}

private onIn() {

this.input.text = parseInt(this.input.text) + '';

}

private onOut() {

var num = parseInt(this.input.text);

this.input.text = num + this.addStr;

//如果不是 stepNum 的整数倍 需要修正

if(num%this.stepNum != 0){

this.nowNum = Math.ceil(num/this.stepNum) \* this.stepNum;

}

MyTool.changeGray(this.addBtn,false,true);

MyTool.changeGray(this.reduceBtn,false,true);

if(num >= this.\_maxNum)

{

this.nowNum = this.\_maxNum;

MyTool.changeGray(this.addBtn,true,true);

}

else if(num <= this.minNum) {

this.nowNum = this.minNum;

MyTool.changeGray(this.reduceBtn,true,true);

}

else

{

this.\_nowNum = num;

}

this.renewCoin();

}

private onChange(){

if(this.input.text == "")

{

this.\_nowNum = 0;

this.renewCoin();

return;

}

var num = parseInt(this.input.text);

//如果不是 stepNum 的整数倍 需要修正

if(num%this.stepNum != 0){

this.nowNum = Math.ceil(num/this.stepNum) \* this.stepNum;

}

if(num > this.\_maxNum)

{

this.nowNum = this.\_maxNum;

}

//else if(num < this.minNum) {

// this.nowNum = this.minNum;

//}

else

this.\_nowNum = num;

this.renewCoin();

}

private renewCoin(){

this.dispatchEventWith(CommonNumInput.RENEW);

}

public set nowNum(v:number){

this.\_nowNum = v;

this.input.text = v+this.addStr;

var num = parseInt(this.input.text);

MyTool.changeGray(this.addBtn,num >= this.\_maxNum,true);

MyTool.changeGray(this.reduceBtn,num <= this.minNum,true);

}

public get nowNum(){

//if(this.input.text == "")

// this.nowNum = 0;

if(this.\_nowNum < this.minNum) {

this.nowNum = this.minNum;

}

return this.\_nowNum;

}

public init(min:number, max:number, step:number, value?:any){

this.minNum = min;

this.maxNum = max;

this.stepNum = step || 1;

this.input.maxChars = String(max).length+1

if(this.addBtn){

MyTool.changeGray(this.addBtn, false, true);

MyTool.changeGray(this.reduceBtn, false, true);

}

if(value !=undefined){

this.nowNum = value;

MyTool.changeGray(this.addBtn, this.nowNum >= this.\_maxNum, true);

MyTool.changeGray(this.reduceBtn, this.nowNum <= this.minNum, true);

}

}

public setEnabled(v:boolean){

this.addBtn.visible = this.reduceBtn.visible = v;

this.input.touchEnabled = v;

}

}

class ConfirmUI extends game.BaseWindow {

public constructor() {

super();

this.skinName = "AlertSkin";

}

private cancelBtn: eui.Button;

private okBtn: eui.Button;

public closeBtn: eui.Button;

private text: eui.Label;

private textIn;

private fun;

private btnWord;

private sp;

public childrenCreated() {

super.childrenCreated();

this.canBGClose = false;

this.addBtnEvent(this.okBtn, this.onClick);

this.addBtnEvent(this.cancelBtn, this.onCancelClick);

this.addBtnEvent(this.closeBtn, this.onCloseClick);

}

public show(v?,fun?,btnWord?,sp?){

this.textIn = v;

this.fun = fun;

this.btnWord = btnWord;

this.sp = sp || {};

super.show();

}

public onShow(){

MyTool.setColorText(this.text,this.textIn);

this.text.validateNow()

if(this.text.numLines > 1 && !this.sp.middle)

this.text.textAlign = 'left'

if(this.btnWord)

{

this.cancelBtn.label = this.btnWord[0];

this.okBtn.label = this.btnWord[1];

}

var ww = GameManager.container.width;

var hh = GameManager.container.height;

this.x = (ww - this.width) / 2;

this.y = (hh - this.height) / 2;

this.closeBtn.visible = false;

}

private onClick(){

this.hide();

if(this.fun)

this.fun(1);

}

private onCancelClick(){

this.hide();

if(this.fun)

this.fun(2);

}

private onCloseClick(){

this.hide();

if(this.fun)

this.fun(3);

}

}

class DebugUI extends game.BaseUI {

private static \_instance:DebugUI;

public static getInstance() {

if (!this.\_instance) this.\_instance = new DebugUI();

return this.\_instance;

}

private backBtn: eui.Button;

private con: eui.Group;

public childrenCreated() {

super.childrenCreated();

this.addBtnEvent(this.backBtn,this.hide)

var wx = window['wx'];

if(!wx)

return;

this.addB('数据库',()=>{

const db = wx.cloud.database()

})

}

private addB(label,fun){

var btn = new eui.Button();

btn.skinName = 'Btn1Skin'

this.con.addChild(btn);

this.addBtnEvent(btn,fun);

}

}

class DownList extends game.BaseContainer {

public static SELECT:string = "SELECT";

private btn: eui.Group;

private img: eui.Image;

private titleText: eui.Label;

private listCon: eui.Group;

private scroller: eui.Scroller;

private list: eui.List;

private openHeight:number;

private isOpen:boolean

public defaultStr="全部"

private dataArr;

public constructor() {

super();

this.skinName = "DownListSkin"

}

public childrenCreated() {

super.childrenCreated();

this.addBtnEvent(this.btn,this.onOpen);

this.scroller.viewport = this.list;

this.list.addEventListener(eui.ItemTapEvent.ITEM\_TAP,this.onSelect,this);

this.list.itemRenderer = DownListItem;

this.openHeight = this.height;

this.isOpen = false;

this.renew();

}

private onOpen(e:egret.TouchEvent){

e.stopImmediatePropagation()

this.isOpen = !this.isOpen;

if(this.isOpen)

{

this.stage.addEventListener(egret.TouchEvent.TOUCH\_TAP,this.onClickStage,this);

}

this.renew();

}

private onClickStage(){

GameManager.stage.removeEventListener(egret.TouchEvent.TOUCH\_TAP,this.onClickStage,this);

this.isOpen = false;

this.renew();

}

private renew(){

if(this.isOpen && !this.listCon.parent)

this.addChild(this.listCon);

else if(!this.isOpen && this.listCon.parent)

this.removeChild(this.listCon);

}

//select，传入data的值.不是index

public setData(arr,select=0)

{

this.dataArr = arr;

this.list.dataProvider = new eui.ArrayCollection(arr);

this.height = Math.min(this.openHeight,arr.length \* 56 - 6 + 60 - 1);

this.selectValue = select;

}

public get selectValue(){

var f= this.list.selectedItem;

if(f) return f.data;

return 0;

}

public set selectValue(select){

var arr = this.dataArr;

for(var i = 0 ; i < arr.length ; i++)

{

if(arr[i].data == select)

{

this.list.selectedIndex = i;

break;

}

}

if(i >= arr.length)

{

select = 0;

this.list.selectedIndex = 0;

}

this.renewBtn(select)

}

private renewBtn(select){

var arr = this.dataArr;

for(var i = 0 ; i < arr.length ; i++)

{

if(select == arr[i].data)

{

this.titleText.text = arr[i].label

this.img.source = arr[i].icon

return;

}

}

this.titleText.text = this.defaultStr;

}

private onSelect(){

var f= this.list.selectedItem;

var data = f? f.data:0

this.dispatchEventWith(DownList.SELECT,true,data);

this.renewBtn(data);

//this.isOpen = false;

//this.renew();

}

}

class DownListItem extends game.BaseItem {

private con: eui.Group;

private img: eui.Image;

private text: eui.Label;

public constructor() {

super();

this.skinName = "DownListItemSkin"

}

public childrenCreated(){

super.childrenCreated();

}

public dataChanged(){

if(this.data.icon)

{

this.img.source = this.data.icon

this.con.addChildAt(this.img,0)

}

else

{

MyTool.removeMC(this.img)

}

this.text.text = this.data ? (""+this.data.label):"全部";

if(this.data.label2)

this.text.text += this.data.label2;

}

}

class LeftList extends game.BaseContainer {

public static SELECT:string = "SELECT";

private btn: eui.Group;

private img: eui.Image;

private listCon: eui.Group;

private scroller: eui.Scroller;

private list: eui.List;

private openHeight:number;

private isOpen:boolean

public defaultStr="全部"

private dataArr;

public constructor() {

super();

this.skinName = "LeftListSkin"

}

public childrenCreated() {

super.childrenCreated();

this.addBtnEvent(this.btn,this.onOpen);

this.scroller.viewport = this.list;

this.list.addEventListener(eui.ItemTapEvent.ITEM\_TAP,this.onSelect,this);

this.list.itemRenderer = DownListItem;

this.openHeight = this.height;

this.isOpen = false;

this.renew();

}

private onOpen(e:egret.TouchEvent){

//e.stopImmediatePropagation()

this.isOpen = !this.isOpen;

if(this.isOpen)

{

egret.callLater(function(){

this.stage.addEventListener(egret.TouchEvent.TOUCH\_TAP,this.onClickStage,this,true);

},this)

}

this.renew();

}

private onClickStage(e){

e.stopImmediatePropagation()

GameManager.stage.removeEventListener(egret.TouchEvent.TOUCH\_TAP,this.onClickStage,this,true);

this.isOpen = false;

this.renew();

}

private renew(){

if(this.isOpen && !this.listCon.parent)

this.addChild(this.listCon);

else if(!this.isOpen && this.listCon.parent)

this.removeChild(this.listCon);

}

//select，传入data的值.不是index

public setData(arr,select=0)

{

this.dataArr = arr;

this.list.dataProvider = new eui.ArrayCollection(arr);

this.listCon.height = arr.length \* 56;

this.selectValue = select;

}

public get selectValue(){

var f= this.list.selectedItem;

if(f) return f.data;

return 0;

}

public set selectValue(select){

var arr = this.dataArr;

for(var i = 0 ; i < arr.length ; i++)

{

if(arr[i].data == select)

{

this.list.selectedIndex = i;

break;

}

}

if(i >= arr.length)

{

select = 0;

this.list.selectedIndex = 0;

}

this.renewBtn(select)

}

private renewBtn(select){

var arr = this.dataArr;

for(var i = 0 ; i < arr.length ; i++)

{

if(select == arr[i].data)

{

//this.titleText.text = arr[i].label

this.img.source = arr[i].icon

return;

}

}

//this.titleText.text = this.defaultStr;

}

private onSelect(){

var f= this.list.selectedItem;

var data = f? f.data:0

this.dispatchEventWith(DownList.SELECT,true,data);

this.renewBtn(data);

//this.isOpen = false;

//this.renew();

}

}

class TipsUI extends game.BaseContainer{

private static instancePool = [];

private static showTips = [];

public static getInstance() {

var instance = this.instancePool.pop()

if (!instance)

instance = new TipsUI();

return instance;

}

private text: eui.Label;

private timer;

public toY

public constructor() {

super();

this.skinName = 'TipsUISkin';

}

public show(v?,cd?){

for(var i=0;i<TipsUI.showTips.length;i++)

{

var item = TipsUI.showTips[i];

item.toY -= 70

egret.Tween.removeTweens(item);

var tw = egret.Tween.get(item);

tw.to({y:item.toY},Math.abs(item.toY - item.y)\*2);

}

TipsUI.showTips.push(this)

egret.clearTimeout(this.timer);

//this.verticalCenter = 0;

GameManager.stage.addChild(this);

MyTool.setColorText(this.text,v);

if(this.text.numLines > 1)

this.text.textAlign = 'left'

this.visible = false

this.timer = egret.setTimeout(this.onTimer,this,cd + (TipsUI.showTips.length-1)\*100);

egret.setTimeout(function(){

this.visible = true

},this,(TipsUI.showTips.length-1)\*100);

this.validateNow();

this.x = (GameManager.stage.stageWidth -this.width)/2

this.y = GameManager.stage.stageHeight \* 0.2;

this.toY = this.y;

}

private onTimer(){

this.hide();

}

public hide(){

egret.clearTimeout(this.timer);

MyTool.removeMC(this);

PopUpManager.testShape();

TipsUI.instancePool.push(this)

var index = TipsUI.showTips.indexOf(this)

if(index != -1)

TipsUI.showTips.splice(index,1)

}

}

class TouchTipsUI extends game.BaseContainer{

private static instance:TouchTipsUI;

public static getInstance() {

if (!this.instance) this.instance = new TouchTipsUI();

return this.instance;

}

private text: eui.Label;

private timer;

public constructor() {

super();

this.skinName = 'TouchTipsSkin';

}

public show(e?,str?){

if(!str)

return;

GameManager.container.addChild(this);

MyTool.setHtml(this.text,str);

this.validateNow();

this.y = e.stageY - this.height - 100;

if(this.y < 0)

{

this.y = e.stageY + 120;

}

this.x = e.stageX - this.width/2;

if(this.x < 10)

{

this.x = 10;

}

else if(this.x + this.width > 630)

{

this.x = 630 - this.width;

}

GameManager.stage.once(egret.TouchEvent.TOUCH\_CANCEL,this.hide,this)

}

public hide(){

MyTool.removeMC(this);

}

}

class UpList extends game.BaseContainer {

public static SELECT:string = "SELECT";

private btn: eui.Group;

private titleText: eui.Label;

private listCon: eui.Group;

private scroller: eui.Scroller;

private list: eui.List;

private openHeight:number;

private isOpen:boolean

public defaultStr="全部"

private dataArr;

public constructor() {

super();

this.skinName = "UpListSkin"

}

public childrenCreated() {

super.childrenCreated();

this.addBtnEvent(this.btn,this.onOpen);

this.scroller.viewport = this.list;

this.list.addEventListener(eui.ItemTapEvent.ITEM\_TAP,this.onSelect,this);

this.list.itemRenderer = DownListItem;

this.openHeight = this.height;

this.isOpen = false;

this.renew();

}

private onOpen(e:egret.TouchEvent){

//e.stopImmediatePropagation()

this.isOpen = !this.isOpen;

if(this.isOpen)

{

egret.callLater(function(){

this.stage.addEventListener(egret.TouchEvent.TOUCH\_TAP,this.onClickStage,this,true);

},this)

}

this.renew();

}

private onClickStage(e){

e.stopImmediatePropagation()

GameManager.stage.removeEventListener(egret.TouchEvent.TOUCH\_TAP,this.onClickStage,this,true);

this.isOpen = false;

this.renew();

}

private renew(){

if(this.isOpen && !this.listCon.parent)

this.addChild(this.listCon);

else if(!this.isOpen && this.listCon.parent)

this.removeChild(this.listCon);

}

//select，传入data的值.不是index

public setData(arr,select=0)

{

this.dataArr = arr;

this.list.dataProvider = new eui.ArrayCollection(arr);

this.listCon.height = arr.length \* 56;

this.selectValue = select;

}

public get selectValue(){

var f= this.list.selectedItem;

if(f) return f.data;

return 0;

}

public set selectValue(select){

var arr = this.dataArr;

for(var i = 0 ; i < arr.length ; i++)

{

if(arr[i].data == select)

{

this.list.selectedIndex = i;

break;

}

}

if(i >= arr.length)

{

select = 0;

this.list.selectedIndex = 0;

}

this.renewBtn(select)

}

private renewBtn(select){

var arr = this.dataArr;

for(var i = 0 ; i < arr.length ; i++)

{

if(select == arr[i].data)

{

this.titleText.text = arr[i].label

return;

}

}

this.titleText.text = this.defaultStr;

}

private onSelect(){

var f= this.list.selectedItem;

var data = f? f.data:0

this.dispatchEventWith(DownList.SELECT,true,data);

this.renewBtn(data);

//this.isOpen = false;

//this.renew();

}

}