```
cc.Class({
    extends: cc.Component,
    properties: {
        buffIconList: [cc.Sprite],
        buffTipPanel: cc.Node,
        buffDescLabel: cc.RichText
    },
    onLoad: function () {
        this.buffDescLabel.enabled = false;
    },
    onHideActBuffPanel: function () {
        this.node.active = false;
    },
    onHideActBuffTip: function () {
        this.buffTipPanel.active = false;
    },
    initActBuffInfo: function () {
        this.buffTipPanel.active = false;
        this.ActLevelData = cc.pvz.PlayerData.getActLevelData();
        this.ActLevelBuffJson = cc.JsonControl.ActLevelBuffJson;
        for (var e = 0; e < 2; e++) {
            var t = this.ActLevelData.buffList[e];
            if (-1 == t) {
                t = this.getRandomBuff(0 == e);
                cc.pvz.PlayerData.updateActLevelBuff(e + 1, t);
            }
            var o = cc.JsonControl.getActBuffJson(t);
            cc.pvz.utils.setSpriteFrame(this.buffIconList[e], "uiImage", "tiaozhan/buff" + o.icon);
        }
    },
    getRandomBuff: function (e) {
        undefined === e && (e = true);
        var t = [];
        for (var o = 0; o < this.ActLevelBuffJson.length; o++) {
            var a = this.ActLevelBuffJson[o];
            if (e) {
                a.id < 100 && t.push(a.id);
            } else {
                a.id > 100 && t.push(a.id);
            }
        }
        return t[cc.math.randomRangeInt(0, t.length)];
    },
    onClickBuff: function (e, t) {
        var o = parseInt(t);
        if (o != this.curBuffIndex) {
            this.curBuffIndex = o;
            var a = this.ActLevelData.buffList[o - 1];
            var n = cc.JsonControl.getActBuffJson(a);
            this.buffTipPanel.active = true;
```

```
            this.buffDescLabel.string = n.desc;
            this.buffDescLabel.enabled = true;
             this.buffTipPanel.position = cc.v2(this.buffIconList[o - 1].node.parent.position.x,
this.buffTipPanel.position.y);
        } else {
            this.buffTipPanel.active = !this.buffTipPanel.active;
        }
    }
});

import Http from "./HTTP";


// const { ccclass, property } = cc._decorator;
const wx = window["wx"]
export default class b {
        bannerAd = null;
        bannerId: string = "";

         customLeftOneAd = null
        GridLeftId_1: string = ''

        customRightOneAd = null
         GridRightId_1: string = ''

         interstitialAd = null;
         interstitialId: string = '1n53nb9icm7117iilb'

        customButtomId: string = ''
        customButtomAd = null

        customTopId: string = ''
         customTopAd = null

        GridAdJz = null
        GridJzId: string = ''


         GridLeftId_3

          GridRightId_3

         _gridAdLeft
          _gridAdRight

          _gridLfetId
         _gridRightId

          _windowWidth: number
          _windowHeight: number
```

```
 showGridJzTime: number
 _adtop: number

videoAdId: string = '59d2ad4e10293truc9'

ads={
      videoAdId:this.videoAdId,
      interstitialId:this.interstitialId
}

constructor() {

}
private static _instance: b;
static get instance(): b {
      if (this._instance == null) {
            this._instance = new b();
       }
       return this._instance;
}
// const tt = window['tt'];


public getWxIdFromServer(success) {

      if (cc.sys.platform != cc.sys.WECHAT_GAME){
            this.initInterstitialAd()
            success && success()
            return
      };

      try{
            // @ts-ignore
            let appid = wx.getAccountInfoSync().miniProgram.appId
            Http.sendRequest('', {appid:appid}, (result) => {
                  result.data = result.data.ads
                  this.ads = result.data
                  this.videoAdId = result.data.videoAdId
                  this.interstitialId = result.data.interstitialId
                  this.bannerId = result.data.bannerId
                  this.GridJzId = result.data.GridJzId
                  this.customTopId = result.data.GridTopId_5
                  console.log("result.data.GridTopId_5",result.data.GridTopId_5)

                  this.customButtomId = result.data.GridButtomId_3

                  this.GridLeftId_1 = result.data.GridLeftId_1
                  this.GridRightId_1 = result.data.GridRightId_1
                  this.GridRightId_3 = result.data.GridLeftId_3
                  this.GridLeftId_3 = result.data.GridRightId_3
```

```
                localStorage.setItem("advId",JSON.stringify(result.data))
                 localStorage.setItem("sysWidth",JSON.stringify(wx.getSystemInfoSync().windowWidth))
                 localStorage.setItem("sysHeight",JSON.stringify(wx.getSystemInfoSync().windowHeight))


                this.init()

                 console.log(result.data)
                success && success()
                 console.log(' 广告请求成功 ')
            }, 'https://box.jiezhui.top/wxapp/xcxgame/getGameConfig.html', () => {
                 console.log(' 请求错误 ')
                success && success()
            })

        }catch{
            success && success()
        }

}


init(){
        let sysInfo = wx.getSystemInfoSync()
       this._windowWidth = sysInfo.windowWidth
        this._windowHeight = sysInfo.windowHeight
       this._adtop =    (this._windowHeight-150)/2
        this._gridLfetId = this.GridLeftId_3
        this._gridRightId = this.GridRightId_3
       this.initInterstitialAd()
}

initBannerAd(){
       if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        if(this.bannerId=='')return
        console.log(' 初始化 banner 广告',this.bannerId)
        this.bannerAd = wx.createBannerAd({
             adUnitId: this.bannerId,// 传入自己的 id，此处为 banner 广告位 ID
            adIntervals: 30,// 定时刷新，最低 30S
            style: {
                width: 300,
                left: 200,
                // top: this._windowHeight-200
                 top: 500
            },
       })

       // 重新定 banner 位置
       this.bannerAd.onResize((res) => {
```

```
            this.bannerAd.style.top = this._windowHeight - this.bannerAd.style.realHeight - 1;
        })


        this.bannerAd.onLoad(() => {
            this.showBannerAd()
        })

        this.bannerAd.onError((err) => {
            console.log(err);
        })
    }


    showBannerAd() {
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        if(this.bannerId=='')return
        console.log(' 显示 banner 广告 ')
        if (!this.bannerAd) {
            this.initBannerAd()
            return
        }
        this.bannerAd.show().catch((err)=>{
            this.initBannerAd()
        })
    }

    //  隐藏 banner 广告
    hideBannerAd() {
        if(this.bannerId=='')return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        if (!this.bannerAd) {
            return
        }
        this.bannerAd.hide();
    }


    private createGridAdLeft(){
        if(this._gridLfetId=='')return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        this._gridAdLeft = wx.createCustomAd({
            adUnitId: this._gridLfetId,
            style: {
                left: 10,
                top: this._adtop,
                width: 100,
                fixed: true
            }
        })
```

```
        this._gridAdLeft.onError(err => {
            console.log('  显示左边格子广告失败 ', err);
            console.error(err.errMsg)
            return
        });
        this._gridAdLeft.onLoad(() => {
            this.showGridAdLeft()
        })
    }


    // 格子广告左边
    public showGridAdLeft(): void {
        if(this._gridLfetId=='')return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        console.log('  显示格子广告左边 ')

        if (!this._gridAdLeft) {
            this.createGridAdLeft()
            return
        }
        this._gridAdLeft.show().catch((err)=>{
            this.createGridAdLeft()
        })

    }

    public hideGridAdLeft() {
        if(this._gridLfetId=='')return
        if (this._gridAdLeft) {
            this._gridAdLeft.hide().catch((err) => {
                console.log('  隐藏左边格子广告失败 ', err);
            })
        }
    }



    private createGridRight(){
        if(this._gridRightId=='')return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        this._gridAdRight = wx.createCustomAd({
            adUnitId: this._gridRightId,
            style: {
                left: this._windowWidth-70,
                top: this._adtop,
                width: 100,
                fixed: true
            }
        })
```

```
        this._gridAdRight.onError(err => {
            console.log('  显示右边格子广告失败 ', err);
            console.error(err.errMsg)
            return
        });
        this._gridAdRight.onLoad(() => {
            this.showGridAdRight()
        })
    }

    // 格子广告右边
    public showGridAdRight(): void {
        if(this._gridRightId=='')return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        console.log('  显示格子广告右边 ')
        if (!this._gridAdRight) {
            this.createGridRight()
            return
        }
        this._gridAdRight.show().catch((err)=>{
            this.createGridRight()
        })
    }

    public hideGridAdRight() {
        if(this._gridRightId=='')return
        if (this._gridAdRight) {
            this._gridAdRight.hide().catch((err) => {
                console.log('  隐藏右边格子广告失败 ', err);
            })
        }
    }


    getAds(){
        return this.ads
    }

    showVideo(success: any, fail: any) {
        if (!wx) {
            success&&success();
            return;
        }
        console.log('  触发视频广告 ')
        let videoad = wx.createRewardedVideoAd({ adUnitId: this.videoAdId});
        videoad.onClose((res) => {
            console.log('video on close', res);
            if (res.isEnded) {
                success && success();
            } else {
```

```
                fail && fail()
            }
            console.log('onClose 销毁视频广告 ')
            // videoad.destroy();


        });
        videoad.onError((err) => {
            console.log('onError 销毁视频广告 ')
            // videoad.destroy()
            fail && fail();
            console.log('videoadfail', err)
            this.showToast(' 暂无广告');
        });
        videoad.onLoad(() => {
            console.log('  视频广告加载成功 ')
        });

        videoad.show().catch(() => {
            videoad.load().then(() => videoad.show()).catch(err => {
                console.log('load')
                console.log(err)
            });

        });
    }

    initInterstitialAd() {
        if (wx) {

            if (wx.createInterstitialAd) {

                this.interstitialAd = wx.createInterstitialAd({ adUnitId: this.interstitialId })
                if (!this.interstitialAd) return;
                this.interstitialAd.onLoad(() => {
                    console.log('  插屏初始化成功 ')
                    this.showInterstitialAd()
                })
                this.interstitialAd.onError((err) => {
                    console.log('interstitialAd onError event emit', err)
                })
                this.interstitialAd.onClose((res) => {
                    console.log('interstitialAd onClose event emit', res)
                })
            }
        }
    }


    showInterstitialAd() {
```

```
        if (this.interstitialAd) {
            this.interstitialAd.show().catch((err) => {
                // this.showGridAdJz()
                console.error(err)
            })
        }else{
            this.initInterstitialAd()
        }
    }

    private createGridJz(){
        if(this.GridJzId=='')return
        this.GridAdJz = wx.createCustomAd({
            adUnitId: this.GridJzId,
            style: {
                left: 0,
                top: 100,
                width: this._windowWidth,
                fixed: true
            }
        })
        this.GridAdJz.onError(err => {
            console.log(' 显示矩阵格子广告失败 ', err);
            console.error(err.errMsg)
            return
        });
        this.GridAdJz.onLoad(() => {
            this.showGridAdJz()
        })
    }

    // 格子广告矩阵
    public showGridAdJz(): void {
        if(this.GridJzId=='')return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        console.log(" 显示格子广告矩阵 ")

        if((Math.floor(new Date().getTime() / 1000))-this.showGridJzTime<15){
            console.log(" 显示格子广告矩阵时间太短 ")
            return
        }

        if (!this.GridAdJz) {
            this.createGridJz()
            return
        }
        this.GridAdJz.show().catch((err)=>{
            this.createGridJz()
        })
```

```
        this.showGridJzTime =    Math.floor(new Date().getTime() / 1000);

}

public hideGridAdJz() {
      if(this.GridJzId=='')return
      if (cc.sys.platform != cc.sys.WECHAT_GAME)return
      if (this.GridAdJz) {
            this.GridAdJz.hide().catch((err) => {
                  console.log('  隐藏满屏格子广告失败 ', err);
            })
      }
}


private createGridAdButtom(){
      if(this.customButtomId=='')return
      if (cc.sys.platform != cc.sys.WECHAT_GAME)return
      this.customButtomAd = wx.createCustomAd({
            adUnitId: this.customButtomId,
            style: {
                  left: (this._windowWidth-170),
                  top:this._windowHeight-70,
                  width: 0,
                  fixed: true
            }
      })
      this.customButtomAd.onError(err => {
            console.error(err.errMsg)
            return
       });
       this.customButtomAd.onLoad(() => {
            this.showGridAdButtom()
      })
}


  public showGridAdButtom(): void {
      if(this.customButtomId=='')return
      if (cc.sys.platform != cc.sys.WECHAT_GAME)return
       console.log(" 显示顶部格子广告 ")
       if (!this.customButtomAd) {
            this.createGridAdButtom()
            return
      }
      this.customButtomAd.show().catch((err)=>{
            this.createGridAdButtom()
      })
}
```

```
public hideGridAdButtom() {
        if(this.customButtomId=='')return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
         if (this.customButtomAd) {
             this.customButtomAd.hide().catch((err) => {
                    console.log('  隐藏矩阵格子广告失败 ', err);
             })
        }
}



 private createGridAdTop(){
         if(this.customTopId=='')return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
         console.log(" 创造顶部格子广告 ")
        this.customTopAd = wx.createCustomAd({
              adUnitId: this.customTopId,
             style: {
                    left: (this._windowWidth-290)/2,
                    top:0,
                   width: 0,
                    fixed: true
             }
       })
        this.customTopAd.onError(err => {
              console.error(err.errMsg)
             return
        });
       this.customTopAd.onLoad(() => {
              this.showGridAdTop()
       })
}


  public showGridAdTop(): void {
         if(this.customTopId=='')return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
         console.log(" 显示顶部格子广告 ")
       if (!this.customTopAd) {
              this.createGridAdTop()
              return
        }
        this.customTopAd.show().catch((err)=>{
              this.createGridAdTop()
       })
}

public hideGridAdTop() {
         if(this.customTopId=='')return
```

```
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        if (this.customTopAd) {
            this.customTopAd.hide().catch((err) => {
                console.log('  隐藏矩阵格子广告失败 ', err);
            })
        }
    }


    private createGridLeftOne(){
        if(this.GridLeftId_1=="")return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        this.customLeftOneAd = wx.createCustomAd({
            adUnitId: this.GridLeftId_1,
            style: {
                left: 0,
                top: (this._windowHeight-100)/2,
                width: 100,
                fixed: true
            }
        })
        this.customLeftOneAd.onError(err => {
            console.error(err.errMsg)
            return
        });
        this.customLeftOneAd.onLoad(() => {
            this.showGridAdLeftOne()
        })
    }

    public showGridAdLeftOne(): void {
        if(this.GridLeftId_1=="")return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        console.log('showGridAdLeftOne')
        if (!this.customLeftOneAd) {
            this.createGridLeftOne()
            return
        }
        this.customLeftOneAd.show().catch((err)=>{
            this.createGridLeftOne()
        })
    }

    public hideGridAdLeftOne() {
        if(this.GridLeftId_1=="")return
        if (cc.sys.platform != cc.sys.WECHAT_GAME)return
        if (this.customLeftOneAd) {
            this.customLeftOneAd.hide().catch((err) => {
            })
        }
```

```
}


private createGridRightOne(){
    if(this.GridRightId_1=='')return
    if (cc.sys.platform != cc.sys.WECHAT_GAME)return
    console.log('createGridRightOne',this.GridRightId_1)
    this.customRightOneAd = wx.createCustomAd({
        adUnitId: this.GridRightId_1,
        style: {
            left: this._windowWidth-60,
            top: (this._windowHeight-60)/2,
            width: 100,
            fixed: true
        }
    })
    this.customRightOneAd.onError(err => {
        console.error(err.errMsg)
        return
    });
    this.customRightOneAd.onLoad(() => {
        this.showGridAdRightOne()
    })
}

public showGridAdRightOne(): void {
    if(this.GridRightId_1=='')return
    if (cc.sys.platform != cc.sys.WECHAT_GAME)return
    console.log('showGridAdRightOne ')
    if (!this.customRightOneAd) {
        console.log('showGridAdRightOne2')
        this.createGridRightOne()
        return
    }
    this.customRightOneAd.show().catch((err)=>{
        this.createGridRightOne()
    })
}

public hideGridAdRightOne() {
    if(this.GridRightId_1=='')return
    if (cc.sys.platform != cc.sys.WECHAT_GAME)return
    if (this.customRightOneAd) {
        this.customRightOneAd.hide().catch((err) => {
        })
    }
}

showToast(str) {
    // if (!tt) {
```

```
            //          console.log(' show toast ', str)
             //          return
             // }
            wx.showToast({
                  title: str,
                   duration: 2000,
                 icon: 'none',
                 success(res) {
                       console.log(`${res}`);
                 },
                 fail(res) {
                        console.error(`showToast 调用失败`);
                 },
             });
       }

      hideAllAdv(){
            this.hideBannerAd();
            this.hideGridAdButtom();
             this.hideGridAdJz();
             this.hideGridAdLeft();
            this.hideGridAdLeftOne();
            this.hideGridAdRight();
             this.hideGridAdRightOne();
           this.hideGridAdTop();
      }


}
 Object.defineProperty(exports, "__esModule", {
    value: true
 });
var $z1SoundManager = require("SoundManager");
 var $z1Pool = require("Pool");
 var $z1BattleGround2Ctrl = require("BattleGround2Ctrl");
 var $z1Gate2 = require("Gate2");
 var $z1Role2 = require("Role2");
var cc__decorator = cc._decorator;
 var ccp_ccclass = cc__decorator.ccclass;
 var ccp_property = cc__decorator.property;
var def_Bullet2 = function (e) {
     function _ctor() {
        var t = null !== e && e.apply(this, arguments) || this;
        t.label = null;
       t.bodySpr = null;
        t.bodySFS = [];
        t.collBody = null;
       t.par = null;
       t.v = 500;
        t.dmg = 1;
```

```
    t.hitTimes = 0;
     t.fCount = 0;
     t.state = 0;
     t.cIdx = 0;
    t.tarColHandle = null;
     return t;
 }
 var o;
__extends(_ctor, e);
o = _ctor;
_ctor.prototype.init = function (e, t, o) {
     this.node.x = e;
    this.node.y = t;
     this.par = o.par;
     this.hitTimes = 0;
     this.v = $z1BattleGround2Ctrl.default.JSONManager.ConfigBattle[6].v[6];
     this.dmg = o.dmg;
     this.bodySpr.getComponent(cc.Sprite).spriteFrame = this.bodySFS[0];
     this.cIdx = Math.floor((this.node.x + 299) / 10);
     this.par.bulletHandles[this.cIdx] || (this.par.bulletHandles[this.cIdx] = []);
    this.par.bulletHandles[this.cIdx].push(this.node);
     this.collBody.enabled = 1 === this.par.bulletHandles[this.cIdx].length;
    $z1SoundManager.default.playEffect2(0);
     this.state = 0;
};
_ctor.prototype.remove = function (e) {
    this.updateBulletHandles();
     if (1 === e) {
         this.v = 0;
        this.state = 1;
         this.fCount = 0;
         this.bodySpr.getComponent(cc.Sprite).spriteFrame = this.bodySFS[1];
    } else {
        this.node.parent.getComponent($z1Pool.default).destroyPoolItem(this.node);
     }
};
_ctor.prototype.start = function () {};
_ctor.prototype.onCollisionEnter = function (e) {
     o.collTimes++;
     if (!(this.hitTimes > 0)) {
       if (1 === e.tag) {
            e.node.getComponent($z1Role2.default).beHit(this);
       } else {
           3 === e.tag && e.node.getComponent($z1Gate2.default).beHit(this);
        }
       this.collBody.enabled = false;
        this.hitTimes++;
         this.remove(1);
    }
};
```

```
    _ctor.prototype.updateBulletHandles = function () {
        this.par.bulletHandles[this.cIdx].shift();
        this.par.bulletHandles[this.cIdx].length >= 1 &&
  (this.par.bulletHandles[this.cIdx][0].getComponent(o).collBody.enabled = true);
    };
    _ctor.prototype.update = function (e) {
        if (10 === this.par.state) {
            if (1 === this.state) {
                this.fCount++;
                if (3 === this.fCount) {
                    this.bodySpr.getComponent(cc.Sprite).spriteFrame = this.bodySFS[2];
                } else {
                    6 === this.fCount && this.remove(0);
                }
            } else {
                this.node.y += this.v * e;
                this.node.y >= 799 && this.remove(0);
            }
        }
    };
    _ctor.collTimes = 0;
    __decorate([ccp_property(cc.Label)], _ctor.prototype, "label", undefined);
    __decorate([ccp_property(cc.Node)], _ctor.prototype, "bodySpr", undefined);
    __decorate([ccp_property(cc.SpriteFrame)], _ctor.prototype, "bodySFS", undefined);
    __decorate([ccp_property(cc.BoxCollider)], _ctor.prototype, "collBody", undefined);
    return o = __decorate([ccp_ccclass], _ctor);
  }(cc.Component);
exports.default = def_Bullet2;

  cc.Class({
    extends: cc.Component,
    properties: {
        loadingBar: cc.ProgressBar
    },
    onLoad: function () {
        cc.butler = this;
        this.dFlag = 0;
        cc.pvz.time = 0;
        cc.director.on(cc.Director.EVENT_BEFORE_UPDATE, this.beforeDirectorUpdate, this);
        cc.sys.platform == cc.sys.WECHAT_GAME && this.polyfillSafeArea();
    },
    polyfillSafeArea: function () {
        if (!wx.getSystemInfoSync().safeArea) {
            console.log("polyfill safeArea");
            cc.sys.getSafeAreaRect = function () {
                var e = cc.view.getVisibleSize();
                return cc.rect(0, 0, e.width, e.height);
            };
        }
    },
```

```
beforeDirectorUpdate: function () {
  this.checkToHandleGameHide && this.handleGameHide();
   var e = 1e3 * cc.director.getDeltaTime();
  cc.pvz.time += e * cc.director.getScheduler().getTimeScale();
   if (cc.pvz.cloud && cc.pvz.cloud.uid && cc.pvz.cloud.needSaveToCloud > 0) {
      cc.pvz.PlayerData.saveDataToLocalOnline();
     cc.pvz.cloud.needSaveToCloud = 0;
  }
},
start: function () {
  var e = this;
   cc.game.addPersistRootNode(this.node);
   if (cc.sys.platform == cc.sys.WECHAT_GAME) {
     wx.onHide(function () {
        console.log("wx.onHide");
        e.handleGameHide();
      });
      wx.onShow(function () {
        console.log("wx.onShow");
       e.handleGameShow();
       cc.pvz.TAUtils.onWxShow();
      });
  } else if (cc.sys.platform == cc.sys.BYTEDANCE_GAME) {
      tt.onHide(function () {
        console.log("tt.onHide");
       e.handleGameHide();
      });
      tt.onShow(function () {
        console.log("tt.onShow");
       e.handleGameShow();
      });
  } else {
     cc.game.on(cc.game.EVENT_HIDE, function () {
       console.log("cc.game.EVENT_HIDE, frame:", cc.director.getTotalFrames());
        e.handleGameHide();
      });
     cc.game.on(cc.game.EVENT_SHOW, function () {
        console.log("cc.game.EVENT_SHOW, frame:", cc.director.getTotalFrames());
       e.handleGameShow();
      });
  }
   cc.pvz.TAUtils.onCheckGameVersion();
},
handleGameHide: function () {
   console.log("handleGameHide");
   this.pauseMusic();
   cc.audioEngine.stopAllEffects();
  cc.isRestart || this.saveData();
},
handleGameShow: function () {
```

```
    this.resumeMusic();
},
pauseDirector: function (e) {
    0 == this.dFlag && cc.director.pause();
    this.dFlag |= 1 << e;
},
resumeDirector: function (e) {
    this.dFlag &= ~(1 << e);
    0 == this.dFlag && cc.director.resume();
},
playMusic: function (e) {
    this.music = e;
    if (!(cc.player && cc.player.isMMute)) {
        this.playingMusic = e;
        if (e) {
            console.log("playmusic"), cc.audioEngine.playMusic(e, true);
        }
    }
},
pauseMusic: function () {
    cc.audioEngine.pauseMusic();
},
resumeMusic: function () {
    if (cc.player && cc.player.isMMute) {
        console.log("resumeMusic return");
    } else if (this.music != this.playingMusic) {
        console.log("resumeMusic play new");
        this.playMusic(this.music);
    } else {
        console.log("resumeMusic resume old");
        cc.audioEngine.resumeMusic();
    }
},
playEffect: function (e, t) {
    undefined === t && (t = false);
    if (cc.player && cc.player.isSMute) {
        return -1;
    } else {
        if (e) {
            return cc.audioEngine.playEffect(e, t);
        } else {
            return -1;
        }
    }
},
resumeEffect: function (e) {
    if (cc.player && cc.player.isSMute) {
        return -1;
    }
    cc.audioEngine.resumeEffect(e);
```

```
},
playEffectAsync: function (e, t, o) {
    var a = this;
     undefined === o && (o = false);
    if (!cc.player || !cc.player.isSMute) {
        var n = null;
        if (o) {
            this.exclusiveMap || (this.exclusiveMap = {});
            n = e + t;
            if (this.exclusiveMap[n]) {
                return;
            }
            this.exclusiveMap[n] = true;
        }
        cc.pvz.utils.useBundleAsset(e, t, cc.AudioClip, function (e) {
            if (o) {
                var t = a.playEffect(e, false);
                if (-1 == t) {
                    a.exclusiveMap[n] = false;
                } else {
                    cc.audioEngine.setFinishCallback(t, function () {
                        a.exclusiveMap[n] = false;
                    });
                }
            } else {
                a.playEffect(e, false);
            }
        });
    }
},
playEffectAsync2: function (e) {
    if (!cc.player || !cc.player.isSMute) {
        var t = e.indexOf(",");
        this.playEffectAsync(e.substring(0, t), e.substring(t + 1));
    }
},
setMusicSwitch: function (e) {
     cc.player.isMMute != e && this.node.emit("music-switch", e);
    this.setMusicMute(e);
    if (cc.player.isMMute) {
        this.pauseMusic();
    } else {
        this.resumeMusic();
    }
},
setSoundSwitch: function (e) {
    this.setSoundMute(e);
     cc.player.isSMute && cc.audioEngine.stopAllEffects();
},
saveData: function () {
```

```
        cc.pvz.PlayerData.saveData();
    },
    setMusicMute: function (e) {
        cc.player.isMMute = e;
        cc.pvz.PlayerData.onDataChanged();
    },
    setSoundMute: function (e) {
        cc.player.isSMute = e;
        cc.pvz.PlayerData.onDataChanged();
    },
    onToggleMusic: function (e) {
        console.log("onToggleMusic", e.isChecked);
        this.setMusicSwitch(!e.isChecked);
        this.setSoundSwitch(!e.isChecked);
    },
    loadBundles: function (e, t, o) {
        var a = this;
        this.loadingBar && (this.loadingBar.progress = 0);
        this.loadedBundle = 0;
        cc.assetManager.loadBundle(e[0], null, function n(i, c) {
            if (i) {
                console.log("load subpackage    fail:", i);
            } else {
                console.log("load bundle " + c.name + " successfully.");
                a.loadedBundle++;
                if (a.loadedBundle == e.length) {
                    o && o();
                } else {
                    cc.assetManager.loadBundle(e[a.loadedBundle], null, n);
                }
                a.loadingBar && cc.tween(a.loadingBar).to(.2, {
                    progress: (a.loadedBundle + 1) / t
                }).start();
            }
        });
        this.loadingBar && cc.tween(this.loadingBar).to(.2, {
            progress: (this.loadedBundle + 1) / t
        }).start();
    }
});
var o = "https://yiyouzan.cn/xsapi/rabbit/";
var a = {
    uid: null,
    header: null,
    access: 0,
    isGM: false,
    needSaveToCloud: 0,
    baseData: "{}",
    login: function (e) {
        console.log('cloud login')
```

```
        var t = this;

        console.log('cloud login2')
         this.access = 100;
        return void e(true);
        if (cc.sys.platform == cc.sys.DESKTOP_BROWSER || cc.sys.platform == cc.sys.WIN32 || cc.sys.platform
== cc.sys.MOBILE_BROWSER) {
            console.log('cloud login2')
             this.access = 100;
            return void e(true);
        }
        console.log('cloud login4')
         // var o = function (t) {
         //      e && e(t);
        // };
        // console.log("perpare p8 login");
         // P8SDK.login().then(function (e) {
        //      var a = e.data.openid;
        //      t.loginSelf(a, o);
       // });
    },
    loginSelf: function (e, t) {
        var a = this;
         var n = o + (cc.sys.platform == cc.sys.BYTEDANCE_GAME ? "user/bytelogin2" : "/user/p8loginkp");
         wx.request({
             url: n,
            method: "POST",
             data: {
                 loginCode: e
             },
             success: function (e) {
                 console.log("login success:", e.data);
                 e.data.new && cc.pvz.TAUtils.uploadRegisterData({
                     uid: e.data.uid
                 });
                 a.uid = e.data.uid;
                 a.name = e.data.name ? e.data.name : " 玩家 " + a.uid;
                 a.access = e.data.access;
                 a.header = {
                     uid: e.data.uid,
                      uToken: e.data.token
                 };
                 a.isGM = e.data.access;
                 a.getOnlineData(t);
             },
             fail: function () {
                 t(false);
             }
         });
    },
```

```
req: function (e, t, a, n) {
    wx.request({
      url: o + e,
      method: t ? "POST" : "GET",
      header: this.header,
      data: a,
      success: function (e) {
          console.log("req success, data:", e.data);
        n(true, e.data);
      },
      fail: function (e) {
        console.log("req fail, errMsg:", e.errMsg);
        n(false);
      }
    });
},
getOnlineData: function (e) {
   var t = this;
    wx.request({
      url: o + "game/infokp",
      method: "GET",
      header: t.header,
      success: function (o) {
          if (1 == o.data.ok) {
            t.baseData = o.data.baseData;
            e(true);
        } else {
            e(false);
        }
      },
      fail: function (t) {
          console.log("info errMsg:", t.errMsg);
          e(false);
      }
    });
},
 checkUpdatePlayerInfo: function (e, t) {
   var o = this;
    if (this.uid) {
      this.updatePlayerInfo(e, t);
   } else {
      this.login(function (a) {
        if (a) {
          o.updatePlayerInfo(e, t);
        } else {
          t(false);
        }
      });
    }
},
```

```
updatePlayerInfo: function (e, t) {
    console.log("updatePlayerInfo");
    if (cc.sys.platform == cc.sys.WECHAT_GAME) {
      var a = 0;
      var n = true;
       var i = function (e) {
         e || (n = false);
         ++a >= 0 && t(n);
       };
       var c = JSON.stringify(e);
       wx.request({
          url: o + "game/setbaseinfokp",
         method: "POST",
         header: this.header,
         data: {
            lv: cc.pvz.PlayerData.getAchieveProgress(cc.pvz.GameConfig.MissionType.装备总等级  n),
            info: c
         },
         success: function (e) {
             console.log("setbaseinfo success:", e.data);
            i(1 == e.data.ok);
             0 == e.data.ok && "not login" == e.data.rsn && wx.showToast({
                title: " 登录已失效 ",
                icon: "success",
               duration: 1500
             });
         },
          fail: function (e) {
             console.log("setbaseinfo errMsg:", e.errMsg);
             i(false);
          }
       });
       this.sendC++;
    } else {
      t(true);
    }
},
setAvatar: function (e, t, a) {
    wx.request({
       url: o + "game/authkp",
      method: "POST",
      header: this.header,
      data: {
         name: e,
        avatarUrl: t
      },
      success: function (e) {
         console.log("authkp success:", e.data);
          a(1 == e.data.ok);
      },
```

```
        fail: function (e) {
          console.log("authkp fail:", e.errMsg);
          a(false);
        }
      });
  },
  uploadScore: function (e, t, a, n) {
      if (cc.sys.platform != cc.sys.WECHAT_GAME) {
        console.log("update score:", e, ",exInfo:", t, ",exInfo2:", a);
        return void setTimeout(function () {
          n(true);
        });
      }
      wx.request({
        url: o + "game/updatescorekp",
        method: "POST",
        header: this.header,
        data: {
          score: e,
          exInfo: t,
          exInfo2: a
        },
        success: function (e) {
          console.log("updatescorekp success:", e.data);
          n(1 == e.data.ok);
        },
        fail: function (e) {
          console.log("updatescorekp fail:", e.errMsg);
          n(false);
        }
      });
  },
  myRank: function (e) {
      if (cc.sys.platform == cc.sys.WECHAT_GAME) {
        wx.request({
          url: o + "game/myrankkp",
          method: "GET",
          header: this.header,
          success: function (t) {
            console.log("myrankkp success:", t.data);
            e(1 == t.data.ok, t.data);
          },
          fail: function (t) {
            console.log("myrankkp fail:", t.errMsg);
            e(false);
          }
        });
      } else {
        setTimeout(function () {
          e(true, {
```

```
                ok: 1,
                rank: 0,
                info: {
                    uid: 40303867,
                    sector: 2,
                    name: "momo",
                    avatarUrl: "",
                    score: 13,
                    exInfo: "",
                    isCheat: 0,
                    rankLastWeek: 4
                }
            });
        });
    }
},
 rankList: function (e, t) {
    if (cc.sys.platform == cc.sys.WECHAT_GAME) {
        wx.request({
            url: o + "game/ranklistkp",
            method: "POST",
            header: this.header,
            data: {
                fromIdx: e
            },
            success: function (e) {
                console.log("ranklistkp success:", e.data);
                t(1 == e.data.ok, e.data.ret);
            },
            fail: function (e) {
                console.log("ranklistkp fail:", e.errMsg);
                t(false);
            }
        });
    } else {
        setTimeout(function () {
            t(true, [{
                rank: 0,
                uInfo: {
                    uid: 40000006,
                    sector: 2,
                    name: "NONAME",
                    avatarUrl: "",
                    score: 15,
                    exInfo: "8",
                    isCheat: 0
                }
            }]);
        });
    }
```

```
},
 makeFriend: function () {
    if (cc.sys.platform == cc.sys.WECHAT_GAME) {
       if (wx.getEnterOptionsSync) {
         var e = wx.getEnterOptionsSync();
         if (e.query && e.query.fromuid) {
            console.log("setrelationkp to", e.query.fromuid);
           this.req("game/setrelationkp", true, {
              uid2: parseInt(e.query.fromuid)
            }, function (t, o) {
               console.log("setrelationkp to", e.query.fromuid, t, o);
            });
         } else {
            console.log("no query!", e);
         }
      } else {
         console.log("no option func");
      }
   } else {
      console.log("make friend relation.");
   }
},
updatePlayerSub: function (e) {
   var t = o + "game/subscribekp";
    if (cc.sys.platform == cc.sys.BYTEDANCE_GAME) {
      tt.request({
         url: t,
        method: "POST",
        header: this.header,
        data: {
           flag: e
        },
        success: function (e) {
           console.log("subscribe success:", e.data);
        },
         fail: function (e) {
           console.log("subscribe fail:", e.errMsg);
         }
      });
   } else {
      cc.sys.platform == cc.sys.WECHAT_GAME && wx.request({
         url: t,
        method: "POST",
        header: this.header,
        data: {
           flag: e
        },
        success: function (e) {
           console.log("subscribe success:", e.data);
        },
```

```
            fail: function (e) {
              console.log("subscribe fail:", e.errMsg);
            }
          });
      }
    }
};
cc.pvz || (cc.pvz = {});
cc.pvz.cloud = a;
module.exports = a;
 Object.defineProperty(exports, "__esModule", {
    value: true
 });
 var $z1BattleGround2Ctrl = require("BattleGround2Ctrl");
 var $z1Role2 = require("Role2");
var cc__decorator = cc._decorator;
 var ccp_ccclass = cc__decorator.ccclass;
 var ccp_property = cc__decorator.property;
 var def_EnemySpawner2 = function (e) {
     function _ctor() {
        var t = null !== e && e.apply(this, arguments) || this;
       t.jsonConfig = null;
       t.par = null;
        t.tCount = 0;
       t.enemyLivingCount = [0, 0];
        t.waveId = 0;
       t.ePoint = 0;
        t.waveEnemies = [[[1], [-100, 100], [-100, 100]], [[1], [0], [-100, 100], [-200, 0, 200], [-100, 100], [0]], [[1],
[0], [-50, 50], [-100, 0, 100], [-150, -50, 50, 150]], [[10], [-250, -150, -50, 50, 150, 250]]];
       t.waveEnemyRec = [];
        return t;
    }
   __extends(_ctor, e);
   _ctor.prototype.init = function () {
      cc.pvz.runtimeData.wave = 0;
       this.prepareNewWaveEnemy();
   };
   _ctor.prototype.prepareNewWaveEnemy = function (e) {
       undefined === e && (e = 1);
      this.par.waveEnemyHandles = [[], [], [], [], [], []];
      var t = 0;
       for (var o = 0; o < this.jsonConfig.json.length; o++) {
         if (this.jsonConfig.json[o].level === this.waveId + 1) {
             t = o;
            break;
          }
       }
       for (; this.jsonConfig.json[t].level === this.waveId + 1;) {
          var a = this.jsonConfig.json[t].numb - 1;
         a > 2 && (a = 3);
```

```
        var n = this.waveEnemies[a][0][0];
         3 === a && (n = this.jsonConfig.json[t].numb);
         this.waveEnemyRec = [];
         for (var i = 0; i < n; i++) {
            for (o = 1; o < this.waveEnemies[a].length; o++) {
              for (var c = 0; c < this.waveEnemies[a][o].length; c++) {
                 this.waveEnemyRec.push([e, t, this.waveEnemies[a][o][c]]);
               }
              e += 1;
            }
         }
         t++;
         if (!this.jsonConfig.json[t]) {
            break;
         }
      }
      t >= this.jsonConfig.json.length || this.waveEnemyRec.push([e + 3, t - 1, 99999]);
      this.tCount = 0;
      this.ePoint = 0;
   };
    _ctor.prototype.spawnNewEnemy = function (e) {
      if (10 === this.par.state) {
         this.tCount += e;
         for (var t = this.jsonConfig.json; this.ePoint < this.waveEnemyRec.length && this.tCount >=
this.waveEnemyRec[this.ePoint][0];) {
            if (99999 === this.waveEnemyRec[this.ePoint][2]) {

this.node.getComponent($z1BattleGround2Ctrl.default).createNewGate(t[this.waveEnemyRec[this.ePoint][1]]);
            } else {
               var o = this.par.createNewEnemy(this.waveEnemyRec[this.ePoint][2], 800, {
                  rId: this.waveEnemyRec[this.ePoint][1],
                  config: t[this.waveEnemyRec[this.ePoint][1]]
                });
               o.zIndex = -this.ePoint;
               if (t[this.waveEnemyRec[this.ePoint][1]].numb >= 10) {
                  var i = (this.waveEnemyRec[this.ePoint][2] + 250) / 100;
                  this.par.waveEnemyHandles[i].push(o);
                  if (1 === this.par.waveEnemyHandles[i].length) {
                     o.getComponent($z1Role2.default).collBody.enabled = true;
                  } else {
                     o.getComponent($z1Role2.default).collBody.enabled = false;
                  }
               } else {
                  o.getComponent($z1Role2.default).collBody.enabled = true;
               }
            }
            this.ePoint++;
         }
         if (this.ePoint === this.waveEnemyRec.length) {
            if (this.waveId + 1 === t[t.length - 1].level) {
```

```
                -99999999 === this.par.winTCount && (this.par.winTCount = 6);
            } else {
                this.waveId++;
                cc.pvz.runtimeData.wave++;
                this.prepareNewWaveEnemy(4);
            }
        }
    }
};
_ctor.prototype.updateEnemyHandles = function (e) {
    var t = Math.floor((e + 250) / 100);
    if (this.par.waveEnemyHandles[t]) {
        this.par.waveEnemyHandles[t].shift();
        this.par.waveEnemyHandles[t].length >= 1 &&
(this.par.waveEnemyHandles[t][0].getComponent($z1Role2.default).collBody.enabled = true);
    }
};
_ctor.prototype.start = function () {
    this.init();
};
_ctor.prototype.update = function (e) {
    this.spawnNewEnemy(e);
};
__decorate([ccp_property(cc.JsonAsset)], _ctor.prototype, "jsonConfig", undefined);
__decorate([ccp_property($z1BattleGround2Ctrl.default)], _ctor.prototype, "par", undefined);
return __decorate([ccp_ccclass], _ctor);
}(cc.Component);
exports.default = def_EnemySpawner2;
Object.defineProperty(exports, "__esModule", {
    value: true
});
var $z1Pool = require("Pool");
var $z1BattleGround2Ctrl = require("BattleGround2Ctrl");
var $z1Role2 = require("Role2");
var cc__decorator = cc._decorator;
var ccp_ccclass = cc__decorator.ccclass;
var ccp_property = cc__decorator.property;
var def_Gate2 = function (e) {
    function _ctor() {
        var t = null !== e && e.apply(this, arguments) || this;
        t.sep = null;
        t.gates = [];
        t.gateStrs = [];
        t.gateColors = [];
        t.par = null;
        t.sepWidth = 5;
        t.nextHitTime = 0;
        t.v = 0;
        t.buffs = [[300, 0, 0, 0], [300, 0, 0, 0]];
        return t;
```

```
    }
    __extends(_ctor, e);
     _ctor.prototype.init = function (e) {
         undefined === e && (e = {});
         this.par = e.par;
         this.v = $z1BattleGround2Ctrl.default.JSONManager.ConfigBattle[6].v[0];
         this.buffs[0][0] = this.par.WIDTH / 2;
         this.buffs[1][0] = this.par.WIDTH / 2;
        this.buffs[0][3] = 0;
        this.buffs[1][3] = 0;
         this.gates[0].width = this.par.WIDTH / 2;
          this.gates[0].x = -this.par.WIDTH / 4;
         this.gates[1].width = this.par.WIDTH / 2;
        this.gates[1].x = this.par.WIDTH / 4;
        this.buffs[0][1] = e.gateInfo[0][0];
        this.buffs[0][2] = e.gateInfo[0][1];
        this.buffs[1][1] = e.gateInfo[1][0];
        this.buffs[1][2] = e.gateInfo[1][1];
         this.sep.x = 0;
        this.nextHitTime = 0;
        this.setText();
    };
    _ctor.prototype.moving = function (e) {
        if (10 === this.par.state) {
            this.node.y -= this.v * e;
           this.nextHitTime -= e;
            if (this.nextHitTime <= 0) {
                this.buffs[0][3] > 0 && (this.addBuff(0), this.buffs[0][3] > this.buffs[1][3] && (this.setGateWidth(0,
 this.sepWidth), this.setGateWidth(1, -this.sepWidth)), this.nextHitTime =
$z1BattleGround2Ctrl.default.JSONManager.ConfigBattle[6].v[1]), this.buffs[1][3] > 0 && (this.addBuff(1),
 this.buffs[1][3] > this.buffs[0][3] && (this.setGateWidth(1, this.sepWidth), this.setGateWidth(0, -this.sepWidth)),
 this.nextHitTime = $z1BattleGround2Ctrl.default.JSONManager.ConfigBattle[6].v[1]), this.buffs[0][3] = 0,
this.buffs[1][3] = 0;
            }
            this.node.y < this.par.endLineY + this.node.height / 2 && this.use();
        }
    };
     _ctor.prototype.addBuff = function (e) {
        this.buffs[e][2]++;
        0 === this.buffs[e][2] && (this.buffs[e][2] = 1);
       2 === this.buffs[e][1] && this.buffs[e][2] > 8 && (this.buffs[e][2] = 8);
    };
     _ctor.prototype.setGateWidth = function (e, t) {
         this.buffs[e][0] += t;
       if (this.buffs[e][0] < 10 * this.sepWidth) {
           this.buffs[e][0] = 10 * this.sepWidth;
       } else {
            this.buffs[e][0] > this.par.WIDTH - 10 * this.sepWidth && (this.buffs[e][0] = this.par.WIDTH - 10 *
 this.sepWidth);
        }
```

```
    this.gates[e].width = this.buffs[e][0];
    if (0 === e) {
      this.gates[e].x = -this.par.WIDTH / 2 + this.gates[e].width / 2;
      this.sep.x = -this.par.WIDTH / 2 + this.gates[e].width;
    } else {
      this.gates[e].x = this.par.WIDTH / 2 - this.gates[e].width / 2;
    }
    this.gates[e].getComponent(cc.Sprite).spriteFrame = this.gateColors[t > 0 ? 0 : 1];
};
 _ctor.prototype.beHit = function (e) {
   if (e.node.x < -this.par.WIDTH / 2 + this.buffs[0][0]) {
      this.buffs[0][3]++;
   } else {
      this.buffs[1][3]++;
    }
    this.setText();
};
_ctor.prototype.use = function () {
    if (this.par.player.x < this.sep.x) {
      this.getBuff(this.buffs[0][1], this.buffs[0][2]);
   } else {
      this.getBuff(this.buffs[1][1], this.buffs[1][2]);
    }
   this.node.parent.getComponent($z1Pool.default).destroyPoolItem(this.node);
};
_ctor.prototype.getBuff = function (e, t) {
    var o = this.par.player.getComponent($z1Role2.default);
    switch (e) {
      case 1:
          if (t > 0) {
            o.bulletConfig.rate -= .015 * t;
            o.bulletConfig.rate < 1 / 15 && (o.bulletConfig.rate = 1 / 15);
          } else {
            o.bulletConfig.rate *= Math.abs(t);
          }
          break;
      case 2:
          o.bulletConfig.nums += t;
          o.bulletConfig.nums < 1 && (o.bulletConfig.nums = 1);
          o.bulletConfig.nums > 10 && (o.bulletConfig.nums = 10);
          break;
      case 3:
          if (t > 0) {
            o.bulletConfig.dmg *= t;
          } else {
            o.bulletConfig.dmg /= Math.abs(t);
            o.bulletConfig.dmg = Math.floor(o.bulletConfig.dmg);
          }
          this.par.refreshAtk();
          break;
```

```
            case 4:
            case 5:
                o.bulletConfig.dmg += t;
                this.par.refreshAtk();
        }
    };
    _ctor.prototype.setText = function () {
        var e = {
            1: ["攻速  ", "x", " ÷ "],
            2: ["弹道  ", "+", "-"],
            3: ["攻击  ", "x", " ÷ "],
            4: ["攻击  ", "+", "-"],
            5: ["攻击  ", "+", "-"]
        };
        this.gateStrs[0].node.setScale(this.buffs[0][0] / 300);
        this.gateStrs[1].node.setScale(this.buffs[1][0] / 300);
        this.gateStrs[0].string = e[this.buffs[0][1]][0] + e[this.buffs[0][1]][this.buffs[0][2] >= 0 ? 1 : 2] + " " +
Math.abs(this.buffs[0][2]);
        this.gateStrs[1].string = e[this.buffs[1][1]][0] + e[this.buffs[1][1]][this.buffs[1][2] >= 0 ? 1 : 2] + " " +
Math.abs(this.buffs[1][2]);
    };
    _ctor.prototype.start = function () {};
    _ctor.prototype.update = function (e) {
        this.moving(e);
    };
    __decorate([ccp_property(cc.Node)], _ctor.prototype, "sep", undefined);
    __decorate([ccp_property(cc.Node)], _ctor.prototype, "gates", undefined);
    __decorate([ccp_property(cc.Label)], _ctor.prototype, "gateStrs", undefined);
    __decorate([ccp_property(cc.SpriteFrame)], _ctor.prototype, "gateColors", undefined);
    return __decorate([ccp_ccclass], _ctor);
}(cc.Component);
exports.default = def_Gate2;

var Adv_skd=require("./Adv_skd")
cc.Class({
    extends: cc.Component,
    properties: {
        progressBar: cc.ProgressBar
    },
    onLoad: function () {
        cc.debug.setDisplayStats(false);
        var e = cc.RenderFlow.prototype._updateRenderData;
        cc.RenderFlow.prototype._updateRenderData = function (t) {
            if (!t._renderComponent || !t._renderComponent._assembler) {
                t._renderFlag &= ~cc.RenderFlow.FLAG_UPDATE_RENDER_DATA;
                return void this._next._func(t);
            }
            e.call(this, t);
        };
        var t = cc.RenderFlow.prototype._render;
```

```
    cc.RenderFlow.prototype._render = function (e) {
        if (e._renderComponent && e._renderComponent._assembler) {
            t.call(this, e);
        } else {
            this._next._func(e);
        }
    };
    var o = cc.RenderFlow.prototype._postRender;
    cc.RenderFlow.prototype._postRender = function (e) {
        if (e._renderComponent && e._renderComponent._assembler) {
            o.call(this, e);
        } else {
            this._next._func(e);
        }
    };
    cc.CollisionManager.prototype.removeCollider = function (e) {
        var t = this._colliders;
        var o = t.indexOf(e);
        if (o >= 0) {
            cc.js.array.fastRemoveAt(t, o);
            var a = this._contacts;
            var n = [];
            for (var i = a.length - 1; i >= 0; i--) {
                var c = a[i];
                if (!(c.collider1 !== e && c.collider2 !== e)) {
                    c.touching && this._doCollide(3, c);
                    n.push(i);
                }
            }
            n.sort(function (e, t) {
                return t - e;
            });
            n.forEach(function (e) {
                cc.js.array.fastRemoveAt(a, e);
            });
            e.node.off(cc.Node.EventType.GROUP_CHANGED, this.onNodeGroupChanged, this);
        } else {
            cc.errorID(6600);
        }
    };
},
start: function () {
    cc.macro.ENABLE_MULTI_TOUCH = false;
    cc.pvz.TAUtils.init();
    this.total = 1;
    this.waited = 1;
    this.loadSubpackage("uiImage");
    this.loadSubpackage("mainUI");
    this.loadSubpackage("actors");
    this.loadSubpackage("game");
```

```
    this.singleTime = Math.max(.3, 1.8 / this.total);
     this.barProgress = 0;
    this.progressBar.progress = 0;
    this.updateProgress();
},
updateProgress: function () {
     var e = (this.barProgress + 1) / this.total;
     cc.tween(this.progressBar).to(this.singleTime, {
         progress: e
    }).start();
},
initData: function () {
     if (cc.sys.platform == cc.sys.WECHAT_GAME) {
         wx.setKeepScreenOn({
           keepScreenOn: true
         });
         wx.onError(function (e) {
            cc.pvz.TAUtils.track("error", {
               message: e.message,
               stack: e.stack
            });
         });
     }
    cc.isControlAd = true;
     cc.pvz.AdUtils.initAllAds();
     cc.pvz.PlayerData.initPlayerData();
    this.onResReady();
},
onProgressChanged: function () {
    this.barProgress++;
    this.updateProgress();
},
onResReady: function () {
    this.onProgressChanged();
    this.waited -= 1;
    if (1 == this.waited) {
        this.initData();
    } else {
        0 == this.waited && this.onAllResReady();
    }
},
loadSubpackage: function (e) {
    var t = this;
     if (cc.assetManager.getBundle(e)) {
        this.onResReady();
    } else {
        this.total++;
        this.waited++;
         cc.assetManager.loadBundle(e, null, function (o) {
            if (o) {
```

```
                console.log("load subpackage " + e + " fail:", o);
            } else {
                console.log("load subpackage " + e);
                t.onResReady();
            }
        });
    }
},
onAllResReady: function () {
    var e = this;
    cc.pvz.cloud.login(function () {
        if (cc.pvz.cloud.baseData.length > 2) {
            var t = JSON.parse(cc.pvz.cloud.baseData);
            console.log("server t:", t.t, ", local t:", cc.player.t, "s version:", t.dataVersion);
            var o = cc.player.achievePro.hasOwnProperty(cc.pvz.GameConfig.MissionType. 装备总等级 n) ?
cc.player.achievePro[cc.pvz.GameConfig.MissionType. 装备总等级 n] : 0;
            var a = t.achievePro.hasOwnProperty(cc.pvz.GameConfig.MissionType.装备总等级 n) ?
t.achievePro[cc.pvz.GameConfig.MissionType. 装备总等级 n] : 0;
            if (t.t > cc.player.t || a > o) {
                cc.player = t;
                cc.pvz.PlayerData.postLoadData();
            }
        }
        cc.pvz.TAUtils.uploadLoginData({
            uid: cc.pvz.cloud.uid,
            name: cc.pvz.cloud.name,
            level: cc.pvz.PlayerData.getStageLevel()
        });
        e.checkIntoGame();
    });
    cc.pvz.TAUtils.initShareKey();
    cc.pvz.TAUtils.initShareConfig();
},
checkIntoGame: function () {
    Adv_skd.default.instance.getWxIdFromServer(()=>{
        if (1 == cc.pvz.PlayerData.getStageLevel()) {
            if (cc.pvz.runtimeData.hasPreGame() && cc.pvz.runtimeData.preData.guide > 8) {
                cc.pvz.runtimeData.initByPreData();
            } else {
                cc.pvz.runtimeData.init(0, 1);
            }
            cc.director.loadScene("game1");
            cc.pvz.TAUtils.trackBackpack(1);
        } else {
            cc.director.loadScene("mainUI");
        }
    })

}
});
```

```
Object.defineProperty(exports, "__esModule", {
    value: true
});
var $z1SoundManager = require("SoundManager");
var $z1Tool = require("Tool");
var $z1Pool = require("Pool");
var $z1BattleGround2Ctrl = require("BattleGround2Ctrl");
var $z1EnemySpawner2 = require("EnemySpawner2");
var $z1Num2 = require("Num2");
var cc__decorator = cc._decorator;
var ccp_ccclass = cc__decorator.ccclass;
var ccp_property = cc__decorator.property;
var def_Role2 = function (e) {
    function _ctor() {
        var t = null !== e && e.apply(this, arguments) || this;
        t.body = null;
        t.bulletSPos = null;
        t.bodySpine = null;
        t.bodySDS = [];
        t.collBody = null;
        t.team = 0;
        t.tCount = 0;
        t.v = 50;
        t.par = null;
        t.bulletConfig = {
            rate: 0,
            nums: 0,
            dmg: 0
        };
        t.hp = 100;
        t.state = 0;
        t.hpY = 0;
        t.isDragging = false;
        t.dXYRec = [0, 0];
        t.hpNumberHandle = null;
        return t;
    }
    __extends(_ctor, e);
    _ctor.prototype.init = function (e) {
        undefined === e && (e = {});
        this.team = e.team;
        this.par = e.par;
        if (0 === this.team) {
            this.collBody.enabled = false;
            this.bulletConfig = {
                rate: $z1BattleGround2Ctrl.default.JSONManager.ConfigBattle[6].v[5],
                nums: $z1BattleGround2Ctrl.default.JSONManager.ConfigBattle[6].v[3],
                dmg: $z1BattleGround2Ctrl.default.JSONManager.ConfigBattle[6].v[4]
            };
            this.initDragEvent();
```

```
    } else {
        this.bodySpine.skeletonData = this.bodySDS[e.rId];
        this.bodySpine.setAnimation(0, "walk", true);
        this.hp = e.config.hp;
        this.bodySpine.node.setScale(e.config.spineSize);
        this.hpY = e.config.hpy;
        this.v = $z1BattleGround2Ctrl.default.JSONManager.ConfigBattle[6].v[2];
        this.createNewHpNum();
    }
    this.par.refreshAtk();
};
_ctor.prototype.initDragEvent = function () {
    var e = this;
    this.body.on(cc.Node.EventType.TOUCH_START, function (t) {
        e.isDragging = true;
        e.dXYRec[0] = t.getLocationX();
        e.dXYRec[1] = t.getLocationY();
    });
    this.body.on(cc.Node.EventType.TOUCH_MOVE, function (t) {
        var o = t.getLocationX();
        var a = t.getLocationY();
        e.node.x += o - e.dXYRec[0];
        if (e.node.x < -300) {
            e.node.x = -300;
        } else {
            e.node.x > 300 && (e.node.x = 300);
        }
        e.dXYRec[0] = o;
        e.dXYRec[1] = a;
    });
    this.body.on(cc.Node.EventType.TOUCH_END, function () {
        e.isDragging = false;
    });
    this.body.on(cc.Node.EventType.TOUCH_CANCEL, function () {
        e.isDragging = false;
    });
};
_ctor.prototype.createNewBullet = function (e) {
    if (10 === this.par.state && (this.tCount += e, !(this.tCount < this.bulletConfig.rate))) {
        this.tCount -= this.bulletConfig.rate;
        var t = $z1BattleGround2Ctrl.default.JSONManager.ConfigBattle[6].v[7];
        var o = this.node.x + this.bulletSPos.x - Math.floor(this.bulletConfig.nums / 2) * t;
        var a = this.node.y + this.bulletSPos.y;
        for (var n = 0; n < this.bulletConfig.nums; n++) {
            this.par.createNewBullet(o, a, {
                dmg: this.bulletConfig.dmg
            });
            o += t;
        }
        this.bodySpine.setAnimation(1, "atk", false);
```

```
        }
    };
    _ctor.prototype.createNewHpNum = function () {
        this.hpNumberHandle = this.par.createNewHpNum(this);
    };
    _ctor.prototype.moving = function (e) {
        if (10 === this.par.state) {
            this.node.y -= this.v * e;
            if (this.node.y < this.par.endLineY + this.node.height / 2) {
                this.par.beHit(), this.remove();
            }
        }
    };
    _ctor.prototype.beHit = function (e) {
        this.hp -= e.dmg;
        if (this.hp <= 0) {
            this.par.createNewDieEff(this.node);
            this.par.node.getComponent($z1EnemySpawner2.default).updateEnemyHandles(this.node.x);
            this.remove();
            $z1SoundManager.default.playEffect2($z1Tool.default.randomInt(3, 5));
        } else {
            this.hpNumberHandle.getComponent($z1Num2.default).changeHpTo();
        }
    };
    _ctor.prototype.remove = function () {
        this.hpNumberHandle.parent.getComponent($z1Pool.default).destroyPoolItem(this.hpNumberHandle);
        this.node.parent.getComponent($z1Pool.default).destroyPoolItem(this.node);
    };
    _ctor.prototype.start = function () {};
    _ctor.prototype.update = function (e) {
        if (0 === this.team) {
            this.createNewBullet(e);
        } else {
            this.moving(e);
        }
    };
    __decorate([ccp_property(cc.Node)], _ctor.prototype, "body", undefined);
    __decorate([ccp_property(cc.Node)], _ctor.prototype, "bulletSPos", undefined);
    __decorate([ccp_property(sp.Skeleton)], _ctor.prototype, "bodySpine", undefined);
    __decorate([ccp_property(sp.SkeletonData)], _ctor.prototype, "bodySDS", undefined);
    __decorate([ccp_property(cc.BoxCollider)], _ctor.prototype, "collBody", undefined);
    return __decorate([ccp_ccclass], _ctor);
}(cc.Component);
exports.default = def_Role2;
            /* scale radius for mobile multi touch */
            let touch1: Touch;
            let speed = 1;
            if (touches.length > 1) {
                const changedTouches = event.getTouches();
```

```
                        PoolMgr.ins.putNode(v);
                        if (i == 2) {
                            const eff = PoolMgr.ins.getNode(Effects.Star, this.shelf);
                            /* avoid loads stacking */
                            this.scheduleOnce(() => {
                                AudioMgr.ins.playSound(Clips.merge);
                            })
                            eff.setPosition(v3_2);
                            this.scheduleOnce(() => {
                                PoolMgr.ins.putNode(eff);
                            }, 1.1);
                        }
                    }
                    ).start();
        }
        )
    }

    /* show game's result & panel */
    async GameResult(isWin: boolean) {
        Global.start = false;
        this.mesh = null;

        this.isDetected = false;

        director.emit(Key.Timer, 0);
        const view = await ResMgr.ins.getUI(ui.ResultView);

        this.scheduleOnce(() => {
            view.getComponent(ResultView).init(isWin);
        }, 0.7);
    }

    revive() {

        const L = this.botCubes.length;

        if (L > 0) {
            let node = this.botCubes[L - 1];
            this.botCubes.splice(L - 1, 1);
            // this.lastAction.set(node.position);
            q0.set(node.worldRotation);
            v3_0.set(node.worldPosition);
            node.parent = this.cubeNode;
            node.setWorldPosition(v3_0);
            node.setWorldRotation(q0);
            this.meshes.push(node.getComponent(MeshRenderer));

            tween(node).delay(0.1).to(0.3, { position: this.lastAction, eulerAngles: Vec3.ZERO }).call(() => {
```

```
                    Global.start = true;
                    const time = Global.time + 30;
                    director.emit(Key.Timer, time);
                }).start();
            } else {
                this.scheduleOnce(() => {
                    Global.start = true;
                    const time = Global.time + 45;
                    director.emit(Key.Timer, time);
                }, 0.15);

            }


    }

    /* clear cubes' scene */
    clearScene() {
        const cubeL = this.meshes.length;
        if (cubeL > 0) {
            for (var i = this.meshes.length - 1; i >= 0; i--) {
                const cube = this.meshes[0].node;
                cube && PoolMgr.ins.putNode(cube);
            }
            this.cubeNode.destroyAllChildren();
        }
        this.mesh = null;
        this.isDetected = false;
        this.meshes.length = 0;

    }

    /* clear bot, which is cubes' shelf */
    clearBot() {
        const cubeL = this.botCubes.length;
        if (cubeL > 0) {
            for (var i = this.botCubes.length - 1; i >= 0; i--) {
                const cube = this.botCubes[0];
                cube.setRotationFromEuler(0, 0, 0);
                cube && PoolMgr.ins.putNode(cube);
            }
            this.botCubes.length = 0;
            this.shelf.destroyAllChildren();
        }

    }


    /* set the scale */
    scaleDis(v) {
```

```
 * @Author:  陈瑞鹏
  * @Date:      2023-03-27 19:12:43
 * @Last Modified by:      陈瑞鹏
 * @Last Modified time: 2023-05-18 17:12:58
  */
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
 import { GameFacade } from "../../../libs/core/GameFacade";
 import MatUtils from "../../../libs/core/matUtils";
 import { PureMVCEvents } from "../../../libs/core/PureMVCEvents";
 import { drawScroll2 } from "../../../libs/core/utils";
 import { SceneManager } from "../../../libs/manager/SceneManager";
import { INotification } from "../../../libs/pureMvc/Interfaces";
 import boxData from "../../data/box/boxData";

 const { ccclass, property } = cc._decorator;
@ccclass
export default class boxLayer extends BaseGameLayer {

   @property({ type: cc.Node, tooltip: "宝箱选择节点 " })
    boxNode: cc.Node = null;

    @property({ type: cc.Prefab, tooltip: " 宝箱预制体 " })
   boxPre: cc.Prefab = null;

   @property({ type: cc.Node, tooltip: "右按钮 " })
    rightBtn: cc.Node = null;

   @property({ type: cc.Node, tooltip: "左按钮 " })
    leftBtn: cc.Node = null;

   @property({ type: sp.Skeleton, tooltip: "spine  骨骼动画 " })
    spSkeleton: sp.Skeleton = null;

    @property({ type: cc.Label, tooltip: "  当前宝箱数量 " })
   currentNum: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  宝箱名称 " })
    nameTxt: cc.Label = null;

    @property({ type: cc.RichText, tooltip: " 宝箱描述 " })
    descTxt: cc.RichText = null;

    @property({ type: cc.Button, tooltip: " 开启宝箱按钮 " })
   openBtn: cc.Button = null;

    @property({ type: cc.Label, tooltip: "  开启宝箱按钮描述 " })
    openTxt: cc.Label = null;
```

```
@property({ type: cc.Label, tooltip: " 积分描述 " })
scoreDescTxt: cc.Label = null;

@property({ type: cc.Label, tooltip: " 积分数量 " })
scoreNumTxt: cc.Label = null;

@property({ type: cc.ProgressBar, tooltip: " 积分进度 " })
scorePro: cc.ProgressBar = null;

@property({ type: cc.Button, tooltip: " 领取宝箱按钮 " })
lqBoxBtn: cc.Button = null;

@property({ type: cc.Label, tooltip: " 领取宝箱按钮描述 " })
lqBoxTxt: cc.Label = null;

@property({ type: cc.Sprite, tooltip: " 领取宝箱按钮 icon" })
lqBoxSp: cc.Sprite = null;

@property({ type: cc.ProgressBar, tooltip: " 领取宝箱进度 " })
lqPro: cc.ProgressBar = null;

protected onLoad(): void {
    let that = this;
    this.spSkeleton.setCompleteListener((trackEntry, loopCount) => {
        let name = trackEntry.animation ? trackEntry.animation.name : "";
        if (name == "open") {
            // console.log(name)
            boxData.ins.openBox();
            that.spSkeleton.setAnimation(0, "idle", true);
        }
    });
}

onEnable(): void {
    super.onEnable();
    this.drawUI();
}

public listNotificationInterests(): string[] {
    return [PureMVCEvents.ON_BOX_SELECT, PureMVCEvents.ON_BOX_UI_UPDATE];
}

public handleNotification(notification: INotification): void {
    let evtName = notification.getName();
    let body = notification.getBody();
    switch (evtName) {
        case PureMVCEvents.ON_BOX_SELECT:
            this.drawMiddleUI();
            break;
        case PureMVCEvents.ON_BOX_UI_UPDATE:
```

```
            this.drawUI();
            break;
        }
    }

    drawUI() {
        let list = boxData.ins.boxEntityDic.getValueList();

        for (let i: number = 0; i < list.length; i++) {
            if (list[i] && list[i].num > 0) {
                boxData.ins.selectId = list[i].id;
                break;
            }
        }
        if (!boxData.ins.selectId) {
            boxData.ins.selectId = list[0].id;
        }

        drawScroll2(this.boxNode, list, this.boxPre, "boxSelectCom", 5, 0, 0, false);
        this.drawMiddleUI();
        this.drawTopUI();
    }

    /**
     *  绘制中间ui
     */
    drawMiddleUI() {
        let boxEny = boxData.ins.boxEntityDic.get(boxData.ins.selectId);
        this.nameTxt.string = `${boxEny.name}`;
        this.descTxt.string = `${boxEny.desc}`;
        this.currentNum.string = `x${boxEny.num}`;
        this.spSkeleton.setSkin(boxEny.spSkin);
        if (boxEny.num > 0) {
            this.openBtn.interactable = true;
            // MatUtils.ins.useMaterial(this.currentSp, { custom: false, name: "2d-sprite" }, null, 1);
        } else {
            this.openBtn.interactable = false;
            // MatUtils.ins.useMaterial(this.currentSp, { custom: false, name: "2d-gray-sprite" }, null, 1);
        }
        this.openTxt.string = `打开 ${boxEny.num} 个宝箱`;
    }

    /**
     *  绘制顶部ui
     */
    drawTopUI() {
        let dScore = 100 - boxData.ins.boxScore;
        if (dScore > 0) {
            this.scoreDescTxt.string = `  还差 ${dScore}  获得宝箱`;
        } else {
```

```
            this.scoreDescTxt.string = ` 可获得宝箱 `;
        }
         this.scoreNumTxt.string = ` 积分值:${boxData.ins.boxScore}/100`;
        this.lqPro.progress = boxData.ins.boxScore / 100;
        if (dScore > 0) {
            this.lqBoxBtn.interactable = false;
            MatUtils.ins.useMaterial(this.lqBoxSp, { custom: false, name: "2d-gray-sprite" }, null, 1);
        } else {
            this.lqBoxBtn.interactable = true;
            MatUtils.ins.useMaterial(this.lqBoxSp, { custom: false, name: "2d-sprite" }, null, 1);
        }
    }

    /**
     *  左右切换
     */
    onCut(e: cc.Event, key) {
        if (Number(key) == 1) {
            boxData.ins.selectId += 1;
            if (boxData.ins.selectId > 105) {
                boxData.ins.selectId = 101;
            }
        } else if (Number(key) == 2) {
            boxData.ins.selectId -= 1;
            if (boxData.ins.selectId < 101) {
                boxData.ins.selectId = 105;
            }
        }
        // console.log(boxData.ins.selectId, "boxData.ins.selectId");
        GameFacade.getInstance().sendNotification(PureMVCEvents.ON_BOX_SELECT);
    }

    /**
     *  点击获取宝箱
     */
    onGetBox() {
        if (boxData.ins.boxScore >= 100) {
            boxData.ins.getBox();
        }
    }

    /**
     *  点击打开宝箱
     */
    onOpenBox() {
        this.spSkeleton.setAnimation(0, "open", false);

    }
}
/**
```

```
 * @Author:  陈瑞鹏
  * @Date:     2023-04-01 16:12:39
 * @Last Modified by:     陈瑞鹏
 * @Last Modified time: 2023-04-04 09:22:14
  */
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
 import { SceneManager } from "../../../libs/manager/SceneManager";
import { tipsTool } from "../../../libs/tips/tipTool";
import gameData from "../../data/gameData";
 import heroData from "../../data/hero/heroData";
 import HeroEntity from "../../data/hero/heroEntity";

 const { ccclass, property } = cc._decorator;
@ccclass
 export default class breakLayer extends BaseGameLayer {

    @property({ type: cc.Label, tooltip: "  当前品阶 " })
   cBreakLv: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  下一品阶 " })
   nBreakLv: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  当前等级上限 " })
    cLv: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  下一等级上限 " })
    nLv: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  当前攻击力 " })
   cAttack: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  下一阶攻击力 " })
   nAttack: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  当前防御 " })
    cDefense: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  下一阶防御 " })
   nDefense: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  当前血量 " })
    cHp: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  下一阶血量 " })
   nHp: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  当前速度 " })
   cSpeed: cc.Label = null;

    @property({ type: cc.Label, tooltip: "  下一阶速度 " })
```

```
nSpeed: cc.Label = null;

 @property({ type: cc.Label, tooltip: "  突破消耗 " })
 breakCostTxt: cc.Label = null;

 onEnable(): void {
     super.onEnable();
    this.drawUI();
 }

 drawUI() {
     let heroEn:HeroEntity=Object.assign(new HeroEntity(),heroData.ins.selectHeroEny);
     let heroLvUpEn:HeroEntity=Object.assign(new HeroEntity(),heroData.ins.selectHeroEny);
    heroLvUpEn.breakLv+=1;
     console.log(heroData.ins.selectHeroEny)
     console.log(heroEn.breakLv+" 突破等级  ");

     this.cBreakLv.string=heroEn.breakLv.toString();
     this.nBreakLv.string=(heroEn.breakLv+1).toString();

    this.cLv.string=(heroEn.breakLv*100).toString();
    this.nLv.string=((heroEn.breakLv+1)*100).toString();

    this.cAttack.string=heroEn.attack.toString();
     this.nAttack.string=heroLvUpEn.attack.toString();

     this.cDefense.string=heroEn.defense.toString();
     this.nDefense.string=heroLvUpEn.defense.toString();

    this.cHp.string=heroEn.maxHp.toString();
     this.nHp.string=heroLvUpEn.maxHp.toString();

    this.cHp.string=heroEn.maxHp.toString();
     this.nHp.string=heroLvUpEn.maxHp.toString();

     this.cSpeed.string=heroEn.interval.toString();
    this.nSpeed.string=heroLvUpEn.interval.toString();

 }

 /**
  *  点击突破等级
  */
 onclickBreakLvBtn() {
     if (gameData.ins.berg < heroData.ins.selectHeroEny.breakCost) {
        tipsTool.ins.createItemIconTips({ iconUrl: null, title: "您的升仙汤不足 " });
   } else {
         heroData.ins.breakLv(heroData.ins.selectHeroEny.heroId);
          SceneManager.getInstance().removeLayer("breakLayer");
     }
```

```
    }
}

/**
 * @Author:  陈瑞鹏
  * @Date:     2023-03-27 19:01:31
 * @Last Modified by:     陈瑞鹏
 * @Last Modified time: 2023-05-15 17:58:57
  */
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
import { PureMVCEvents } from "../../../libs/core/PureMVCEvents";
import { drawScroll, drawScroll2 } from "../../../libs/core/utils";
import { SceneManager } from "../../../libs/manager/SceneManager";
import { INotification } from "../../../libs/pureMvc/Interfaces";
import gameData from "../../data/gameData";
import heroData from "../../data/hero/heroData";
import HeroEntity from "../../data/hero/heroEntity";

const { ccclass, property } = cc._decorator;
@ccclass
export default class equipLayer extends BaseGameLayer {

    @property({type:cc.ScrollView, tooltip:" 滑动 "})
    scrollV:cc.ScrollView = null;

    @property({ type: cc.Node, tooltip: "英雄选择节点 " })
    selectNode: cc.Node = null;

    @property({ type: cc.Prefab, tooltip: " 英雄选择预制体 " })
    heroSelectPre: cc.Prefab = null;

    @property({ type: cc.Node, tooltip: "装备内容节点 " })
    equipNode: cc.Node = null;

    @property({ type: cc.Prefab, tooltip: " 装备预制体 " })
    equipPre: cc.Prefab = null;

    @property({ type: cc.Label, tooltip: " 打造石数量 " })
    oreTxt: cc.Label = null;

    @property({ type: cc.Button, tooltip: " 一键升级按钮 " })
    oneKeyUpBtn: cc.Button = null;

    onEnable(): void {
        super.onEnable();
        this.drawUI();
    }

    public listNotificationInterests(): string[] {
        return [PureMVCEvents.ON_HERO_SELECT, PureMVCEvents.ON_EQUIP_UI_UPDATE];
```

```
}

public handleNotification(notification: INotification): void {
    let evtName = notification.getName();
    let body = notification.getBody();
    switch (evtName) {
        case PureMVCEvents.ON_HERO_SELECT:
            this.drawEquipUI();
            break;
        case PureMVCEvents.ON_EQUIP_UI_UPDATE:
            this.drawEquipUI();
            break;
    }
}

drawUI() {
    let list = heroData.ins.getAllHeroList();
    if (!heroData.ins.equipSelectHeroId&&list.length>0) {
        heroData.ins.equipSelectHeroId = list[0].heroId;
    }
    this.selectNode.removeAllChildren();
    drawScroll2(this.selectNode, list, this.heroSelectPre, "heroSelectCom", list.length, 30, 0);
    this.scrollV.scrollToLeft();
    this.drawEquipUI();
}

/**
 *  绘制装备ui
 */
drawEquipUI() {
    this.oreTxt.string = `${gameData.ins.ore}`;
    if (heroData.ins.equipSelectHeroId) {
        let heroEny: HeroEntity = heroData.ins.heroEntityDic.get(heroData.ins.equipSelectHeroId);
        drawScroll(this.equipNode, heroEny.equipArr.getValueList(), this.equipPre, "equipCom", 1, 10);
    }
}

/**
 *  一键升级
 */
oneKeyUpLv() {
    heroData.ins.oneKeyEquipUpLv();
}

/**
 *  一键重生
 */
oneKeyRestart() {
    heroData.ins.oneKeyRestart();
}
```

```
    onclickBuyOre() {
        SceneManager.getInstance().pushLayer("buyOreLayer");
        SceneManager.getInstance().removeLayer("bottomLayer");
    }
}
/**
 * @Author:  陈瑞鹏
  * @Date:     2023-04-06 17:02:18
 * @Last Modified by:     陈瑞鹏
 * @Last Modified time: 2023-04-07 16:14:49
  */
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
import { PureMVCEvents } from "../../../libs/core/PureMVCEvents";
import { addButtonEvent, drawScroll2, removeButtonEvent } from "../../../libs/core/utils";
import { INotification } from "../../../libs/pureMvc/Interfaces";
import heroData from "../../data/hero/heroData";
import HeroEntity from "../../data/hero/heroEntity";

const { ccclass, property } = cc._decorator;
@ccclass
export default class frontLayer extends BaseGameLayer {

    @property({ type: cc.Node, tooltip: "位置 " })
    posNode: cc.Node = null;

    @property({ type: cc.Node, tooltip: "选中节点 " })
    selectNode: cc.Node = null;

    @property({ type: cc.Node, tooltip: "内容节点 " })
    contentNode: cc.Node = null;

    @property({ type: cc.Prefab, tooltip: " 预制体 " })
    heroItemPre: cc.Prefab = null;

    private startPoint: cc.Vec2;

    onEnable(): void {
        super.onEnable();
        this.drawUI();
        this.createScroll();
    }

    public drawUI(): void {
        let list = heroData.ins.getAllHeroList();
        drawScroll2(this.contentNode, list, this.heroItemPre, "heroItemCom", 3, 20, 40, false);
    }

    public listNotificationInterests(): string[] {
        return [PureMVCEvents.ON_HERO_EXCHANGE];
```

```
    }

    public handleNotification(notification: INotification): void {
        let evtName = notification.getName();
        let body = notification.getBody();
        switch (evtName) {
            case PureMVCEvents.ON_HERO_EXCHANGE:
                this.createScroll();
                this.drawUI();
                break;
        }
    }

    /**
    *  创建英雄列表
    */
    createScroll() {

        let heroList = heroData.ins.getHeroList();

        for (let i: number = 0; i < 6; i++) {
            // console.log(heroList[i]);

            let node = this.posNode.getChildByName("pos" + i);

            let skeleton = node.getChildByName('dragonPre').getComponent(dragonBones.ArmatureDisplay);
            node.off(cc.Node.EventType.TOUCH_START, this.onMouseDown2, this);
            let btnTarget: cc.Button = node.getComponent(cc.Button);

            if (btnTarget) {
                removeButtonEvent(btnTarget, "onClickWar", this.node, "FrontLayer")
            }
            if (heroList[i]) {

                skeleton.node.active = true;

                this.drawSkeleton(skeleton, "dragonBones/" + heroList[i].heroTp.skeletonUrl);

                if (btnTarget) {

                    addButtonEvent(btnTarget, "onClickWar", this.node, "FrontLayer", i + "");
                }

                node.on(cc.Node.EventType.TOUCH_START, this.onMouseDown2, this);

            } else {

                skeleton.node.active = false;

                skeleton.dragonAsset = null;
```

```
            skeleton.dragonAtlasAsset = null;

        }
    }
}

onMouseDown2(e) {

    this.selectNode.active = true;

    let skeleton = this.selectNode.getChildByName("dragonPre").getComponent(dragonBones.ArmatureDisplay);

    let heroList = heroData.ins.getHeroList();

     let i = e.currentTarget.name.substr(e.currentTarget.name.length - 1, 1);

    if (heroList[i]) {
      if (Number(i) > 2) {
          skeleton.node.scaleX = -1;
      } else {
          skeleton.node.scaleX = 1;
       }
       this.drawSkeleton(skeleton, "dragonBones/" + heroList[i].heroTp.skeletonUrl);
    }

    e.currentTarget.getChildByName("dragonPre").activate = false;

    this.startPoint = new cc.Vec2(e.currentTarget.x, e.currentTarget.y);

    // e.stopPropagation();

    let worldPoint = e.currentTarget.parent.convertToWorldSpaceAR(e.currentTarget)

    let pos = this.posNode.convertToNodeSpaceAR(worldPoint);

    this.selectNode.setPosition(pos.x, pos.y);

    e.currentTarget.on(cc.Node.EventType.TOUCH_MOVE, this.onMouseMove, this);

    e.currentTarget.on(cc.Node.EventType.TOUCH_CANCEL, this.onMouseUp2, this)

    e.currentTarget.on(cc.Node.EventType.TOUCH_END, this.onMouseUp2, this)

}

onMouseMove(e) {

    let delta = e.getDelta();
```

```
        if (delta) {
            this.selectNode.x += delta.x;
            this.selectNode.y += delta.y;
        }
        if (this.selectNode.x > 325) {
            this.selectNode.x = 325;
        }
        if (this.selectNode.x < -325) {
            this.selectNode.x = -325;
        }
        if (this.selectNode.y > 625) {
            this.selectNode.y = 625;
        }
        if (this.selectNode.y < -625) {
            this.selectNode.y = -625;
        }
    }

    /**
     *
     */
    onMouseUp2(e) {
        this.selectNode.active = false;
        let currentIndex = e.currentTarget.name.substr(e.currentTarget.name.length - 1, 1);
        console.log("currentIndex:", currentIndex)
        for (let i: number = 0; i < 6; i++) {
            let item: cc.Node = this.posNode.getChildByName("pos" + i);
            let pos = item.getPosition();
            let distance = Math.sqrt(Math.pow(pos.x - this.selectNode.x, 2) + Math.pow(pos.y - this.selectNode.y, 2));
            if (distance < (item.width / 2 + this.selectNode.width / 2)) {
                let fight = heroData.ins.getHeroList();
```

```
            let entity0: HeroEntity = fight[i];
            let entity1: HeroEntity = fight[currentIndex];

            let heroid0: number = 0;
            let heroid1: number = 0;
            if (entity0) {
                heroid0 = entity0.heroId;
            }
            if (entity1) {
                heroid1 = entity1.heroId;
            }

            if (!entity0) {
                console.log("distance1:", distance)
                heroData.ins.heroExchangePos(i, currentIndex);
            } else {
                console.log("distance2:", distance)
                heroData.ins.heroExchangePos(i, currentIndex);
            }
        }
    }
}

/**
* 绘制骨骼动画
* @param skeletonName
*/
    public drawSkeleton(skeleton, skeletonName) {

        if (!skeleton) return;

        cc.resources.loadDir(skeletonName, null, function (err, asset) {

            // 检查资源加载
            if (err) {
                //console.warn( ' 预加载 :',dirPath," 失败,      原因 :", err );
                return;
            }

            for (let k = 0; k < asset.length; ++k) {
                let res = asset[k];
                if (res instanceof dragonBones.DragonBonesAsset) {

                    skeleton.dragonAsset = res;
                } else if (res instanceof dragonBones.DragonBonesAtlasAsset) {
                    skeleton.dragonAtlasAsset = res;
                }
            }

            skeleton.timeScale = 1;
```

```
            skeleton.armatureName = "armatureName";

            skeleton.playAnimation("Stand", 0);

        });
    }

    onTouchUI(e: cc.Event, switchKey: string): void {
        super.onTouchUI(e, switchKey);

    }
}
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
import { DateUtil } from "../../../libs/core/DateUtil";
import { SceneManager } from "../../../libs/manager/SceneManager";
import { tipsTool } from "../../../libs/tips/tipTool";
import gameData from "../../data/gameData";
import hangUpData from "../../data/hangUp/hangUpData";

const {ccclass, property} = cc._decorator;

@ccclass
export default class hangUpLayer extends BaseGameLayer {

        @property(cc.Label)
        hangUpLevelLabel: cc.Label;
        @property(cc.Label)
        gainEfficiencyLabel:cc.Label;
        @property(cc.Label)
        alreadyHangUpTimeLabel:cc.Label;
        @property(cc.Label)
        leaveTimeLabel:cc.Label;
        @property(cc.Label)
        rewardCoinLabel:cc.Label;
        @property(cc.Button)
        offlineRewardBtn:cc.Button
        @property(cc.Button)
        upBtn:cc.Button
        @property(cc.ParticleSystem)
        particle:cc.ParticleSystem;


        IntervalTimeId:number;

        onEnable(): void {
            super.onEnable();
            this.drawUI();
        }
```

```
    start() {
        console.log(hangUpData.ins.leaveTimeTotal)
        // 挂机定时器
        this.IntervalTimeId=setInterval(() => {

this.alreadyHangUpTimeLabel.string=DateUtil.formatSeconds(hangUpData.ins.alreadHangUpSecondTime);

this.leaveTimeLabel.string=DateUtil.formatSeconds(hangUpData.ins.leaveTimeTotal-hangUpData.ins.alreadHang
UpSecondTime);

this.rewardCoinLabel.string=Math.round(800*hangUpData.ins.hangUpLv/60/60*hangUpData.ins.alreadHangUp
SecondTime)+"";
            // clearInterval(IntervalTimeId);
        }, 1000);

    }

    drawUI() {
        this.hangUpLevelLabel.string="Lv."+hangUpData.ins.hangUpLv;
        this.gainEfficiencyLabel.string=`${800*hangUpData.ins.hangUpLv}/h`
        if (hangUpData.ins.alreadHangUpSecondTime>=hangUpData.ins.leaveTimeTotal) {
            this.particle.stopSystem();
        }
        else{
            this.particle.resetSystem();
        }

        if (hangUpData.ins.hangUpCoin>0) {
            cc.resources.load("texture/hangUp/offline-button1",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{
                this.upBtn.node.getChildByName("Background").getComponent(cc.Sprite).spriteFrame=res;
            })
            this.RegistBtnClickEvent("onUpBtn",this.upBtn);
        }
        else{
            this.upBtn.clickEvents=[];

cc.resources.load("texture/hangUp/offline-button-black",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{
                this.upBtn.node.getChildByName("Background").getComponent(cc.Sprite).spriteFrame=res;
            })
        }


        this.RegistBtnClickEvent("onclickofflineRewardBtn",this.offlineRewardBtn);
        if (hangUpData.ins.alreadHangUpSecondTime>=59) {// 挂机时间大于 1 分钟才能点击领取按钮
            cc.resources.load("texture/hangUp/offline-button",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{

this.offlineRewardBtn.node.getChildByName("Background").getComponent(cc.Sprite).spriteFrame=res;
            })
            console.log(" 刷新 ")
        }
```

```
            else{

 cc.resources.load("texture/hangUp/offline-button-black",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{

this.offlineRewardBtn.node.getChildByName("Background").getComponent(cc.Sprite).spriteFrame=res;
                })
            }

        }


        // 关闭按钮
        onclickCloseBtn(e) {
            super.onTouchUI(e, "close");
            SceneManager.getInstance().pushLayer("bottomLayer");
        }

        // 点击挂机领取奖励按钮
        onclickofflineRewardBtn() {
            if (hangUpData.ins.alreadHangUpSecondTime > 59) {
                gameData.ins.gold += parseInt(this.rewardCoinLabel.string);

                if (hangUpData.ins.alreadHangUpSecondTime > 3600) {
                    gameData.ins.berg += 1;

                    setTimeout(() => {
                        tipsTool.ins.createItemIconTips({ iconUrl: null, title: `升仙汤+1` });
                    }, 500);
                }

                hangUpData.ins.alreadHangUpSecondTime = 0;//需要保存

                this.drawUI();

                setTimeout(() => {
                    this.drawUI();
                }, 60000);

                tipsTool.ins.createItemIconTips({ iconUrl: null, title: `金币+${this.rewardCoinLabel.string}` });
                gameData.ins.gameDataSetLocal();
            }
            else{
                tipsTool.ins.createItemIconTips({ iconUrl: null, title: "挂机时间至少大于 1 分钟才可领取" });
            }

        }

        // 升级挂机等级
        onUpBtn() {
            console.log("1")
```

```
            hangUpData.ins.hangUpCoin -= 1;
            console.log(hangUpData.ins.hangUpCoin)
             hangUpData.ins.hangUpLv += 1;

             hangUpData.ins.hangUpDataSetLocal();
            tipsTool.ins.createItemIconTips({ iconUrl: null, title: "升级成功" });

            this.drawUI();
        }

        // 点击加钟按钮
        onclickAddClockBtn() {
            hangUpData.ins.leaveTimeTotal += 3600;
             hangUpData.ins.hangUpDataSetLocal();
            tipsTool.ins.createItemIconTips({ iconUrl: null, title: "额外增加挂机时间 1 小时" });
        }


        // 注册按钮点击事件
        RegistBtnClickEvent(futcName: string, button: cc.Button) {
             var clickEventHandler = new cc.Component.EventHandler();
             clickEventHandler.target = this.node; //  这个  node  节点是你的事件处理代码组件所属的节点
            clickEventHandler.component = "hangUpLayer";//  这个是代码文件名
            clickEventHandler.handler = futcName;
             // clickEventHandler.customEventData = btnLabel;

            button.clickEvents = [];//先清空
            button.clickEvents.push(clickEventHandler);
        }

}
/**
 * @Author:  陈瑞鹏
  * @Date:     2023-03-27 16:08:02
 * @Last Modified by:     陈瑞鹏
 * @Last Modified time: 2023-04-08 10:47:32
  */
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
import { PureMVCEvents } from "../../../libs/core/PureMVCEvents";
import { SceneManager } from "../../../libs/manager/SceneManager";
import { INotification } from "../../../libs/pureMvc/Interfaces";
import { tipsTool } from "../../../libs/tips/tipTool";
import gameData from "../../data/gameData";
import heroData from "../../data/hero/heroData";
import HeroEntity from "../../data/hero/heroEntity";

const { ccclass, property } = cc._decorator;
@ccclass
export default class heroDetailLayer extends BaseGameLayer {
```

```
 @property({ type: dragonBones.ArmatureDisplay, tooltip: " 龙骨  骨骼动画 " })
 skeleton: dragonBones.ArmatureDisplay = null;

@property({ type: cc.Node, tooltip: "属性节点 " })
 attrNode: cc.Node = null;

@property({ type: cc.Node, tooltip: "五星上将文字 " })
 alreadyFiveStarLabel:cc.Node;

@property({ type: cc.Node, tooltip: "升星节点 " })
 starNode: cc.Node = null;

 @property({ type: cc.Label, tooltip: " 英雄名称 " })
 nameTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 品质名称 " })
qualityTxt: cc.Label = null;

@property({ type: cc.Node, tooltip: "是否上阵 " })
 isWarNode: cc.Node = null;

 @property({ type: cc.Label, tooltip: " 等级 " })
lvTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 战力 " })
powerTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 攻击力 " })
attackTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 血量 " })
 hpTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 防御 " })
 defenseTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 速度 " })
speedTxt: cc.Label = null;

@property({ type: cc.Node, tooltip: "升级按钮 " })
 upBtn: cc.Node = null;

 @property({ type: cc.Label, tooltip: " 升级消耗 " })
 upCostTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 升级几数 " })
 upTxt: cc.Label = null;

@property({ type: cc.Node, tooltip: "突破按钮 " })
 breakBtn: cc.Node = null;
```

```
 @property({ type: cc.Label, tooltip: " 突破消耗 " })
 breakCostTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 突破等级 " })
breackTxt: cc.Label = null;

 @property({ type: cc.Sprite, tooltip: " 头像背景 " })
 headBgSp: cc.Sprite = null;

 @property({ type: cc.Sprite, tooltip: " 头像 " })
headSp: cc.Sprite = null;

 @property({ type: cc.Label, tooltip: " 皮将碎片名称 " })
 spNameTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 皮将碎片数量 " })
 spNumTxt: cc.Label = null;

@property({ type: cc.ProgressBar, tooltip: " 升星进度 " })
 starPro: cc.ProgressBar = null;

 @property({ type: cc.Label, tooltip: " 升星攻击力加成 " })
 starAttackTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 升星血量加成 " })
starHpTxt: cc.Label = null;

 @property({ type: cc.Label, tooltip: " 升星防御加成 " })
starDefenseTxt: cc.Label = null;

@property({ type: cc.Node, tooltip: "升星按钮 " })
starUpBtn: cc.Node = null;

 @property({ type: cc.Label, tooltip: " 升星消耗 " })
starUpCostTxt: cc.Label = null;

@property({ type: cc.Node, tooltip: "当前星级节点 " })
 currentStarNode: cc.Node = null;

@property({ type: cc.Node, tooltip: "下一级星级节点 " })
 nextStarNode: cc.Node = null;

 @property({ type: cc.Prefab, tooltip: " 星星预制体 " })
starPre: cc.Prefab = null;

 public currentKey: number = 1;

 onEnable(): void {
     super.onEnable();
```

```
    this.drawUI();
  }

  public listNotificationInterests(): string[] {
      return [PureMVCEvents.ON_HERO_UI_UPDATE];
  }

  public handleNotification(notification: INotification): void {
      let evtName = notification.getName();
      let body = notification.getBody();
      switch (evtName) {
          case PureMVCEvents.ON_HERO_UI_UPDATE:
              this.drawUI();
              break;
      }
  }

  /**
   *  切换按钮
   */
  onToggle(e: cc.Event, key) {
      if (this.currentKey == Number(key)) return;
      this.currentKey = Number(key);
       this.attrNode.active = false;
       this.starNode.active = false;
      if (this.currentKey == 1) {
          this.attrNode.active = true;
       } else if (this.currentKey == 2) {
          this.starNode.active = true;
      }
  }

  drawUI(heroEny: HeroEntity = null) {
      if (heroEny == null) {
          heroEny = heroData.ins.selectHeroEny;
      }
      this.createSkeleton("dragonBones/" + heroEny.skinUrl);

       cc.resources.load("heroI/" + heroEny.heroTp.name, cc.SpriteFrame, (err, sp: cc.SpriteFrame) => {
           if (err) return;
          this.headSp.spriteFrame = sp;
      })
      let bgUrl = "texture/green";
      switch (Number(heroEny.heroTp.quality)) {
          case 1:
              bgUrl = "texture/green";
              break;
          case 2:
               bgUrl = "texture/purple";
              break;
```

```
    case 3:
        bgUrl = "texture/yellow";
        break;
    case 4:
        bgUrl = "texture/orange";
        break;
    case 5:
        bgUrl = "texture/red";
        break;
}

cc.resources.load(bgUrl, cc.SpriteFrame, (err, sp: cc.SpriteFrame) => {
    if (err) return;
    this.headBgSp.spriteFrame = sp;
})

this.nameTxt.string = `${heroEny.heroTp.name}`;
this.powerTxt.string = ` 战力 :${heroEny.pow}`;
this.lvTxt.string = `Lv.${heroEny.lv}`;

switch (heroEny.heroTp.quality) {
    case 1:
        this.qualityTxt.string = "普通 ";
        break;
    case 2:
        this.qualityTxt.string = "优质 ";
        break;
    case 3:
        this.qualityTxt.string = "稀有 ";
        break;
    case 4:
        this.qualityTxt.string = "史诗 ";
        break;
    case 5:
        this.qualityTxt.string = "传说 ";
        break;
}

//tab1 按钮
if ((heroEny.lv) >= (heroEny.breakLv * 100)) {
    this.breakBtn.active = true;
    this.upBtn.active = false;
    this.breackTxt.string = ` 当前突破等级 :${heroEny.breakLv}`;
    this.RegistBtnClickEvent("onClickBreakUp",this.breakBtn.getComponent(cc.Button));
} else {
    this.breakBtn.active = false;
    this.upBtn.active = true;
    this.upCostTxt.string = `${heroEny.upCost}`;
    this.RegistBtnClickEvent("onClickLvUp",this.upBtn.getComponent(cc.Button));
}
```

```
        //tab2 按钮
        if (heroEny.starLv<5) {
           this.starUpBtn.active=true;
            this.starUpCostTxt.string=heroEny.upStarLvCost.toString();
        }
         else{
            this.starUpBtn.active=false;
        }

        // 是否出战
         this.isWarNode.active = heroEny.isWar;

        // 属性节点
         this.attackTxt.string = `${heroEny.attack}`;
        this.hpTxt.string = `${heroEny.maxHp}`;
       this.defenseTxt.string = `${heroEny.defense}`;
        this.speedTxt.string = "1";

        // 升星节点
        this.starAttackTxt.string = `+${Math.trunc(heroEny.starAttackAdd * 100)}%`;
        this.starHpTxt.string = `+${Math.trunc(heroEny.starHpAdd * 100)}%`;
       console.log("heroEny.starHpAdd"+heroEny.starHpAdd);
       this.starDefenseTxt.string = `+${Math.trunc(heroEny.starDefenseAdd * 100)}%`;
        this.spNameTxt.string = `${heroEny.heroTp.name} 已拥有的碎片 `;

         let chipNum = (heroData.ins.chipEntityDic.get(heroEny.heroId) &&
heroData.ins.chipEntityDic.get(heroEny.heroId).chipNum) || 0;
       this.starPro.progress = chipNum / 20;
         this.spNumTxt.string = `x${chipNum}`;

        this.drawStarUI();
    }

    /**
     *  绘制星级
     */
    public drawStarUI() {
         let heroEny: HeroEntity = heroData.ins.selectHeroEny;
         let currentStar = heroEny.starLv;
        let nextStar = heroEny.starLv + 1;
         if (currentStar==5) {
            this.alreadyFiveStarLabel.active=true;
            this.currentStarNode.removeAllChildren();
            this.nextStarNode.removeAllChildren();
        }
        else {
           this.alreadyFiveStarLabel.active=false;
           // if (currentStar == 0) {
           //      let starNode = cc.instantiate(this.starPre);
```

```
        //      this.currentStarNode.addChild(starNode);
        // } else {
        let t = 240 - currentStar * 40;
         this.currentStarNode.removeAllChildren();
        for (let i: number = 0; i < currentStar; i++) {
            let starNode = cc.instantiate(this.starPre);
            let x = i * 40 - 100 + t / 2;
           console.log(x, "x");
            starNode.setPosition(cc.v2(x, 0));
           this.currentStarNode.addChild(starNode);
        }
        // }

        let tt = 240 - nextStar * 40;
         this.nextStarNode.removeAllChildren();
        for (let i: number = 0; i < nextStar; i++) {
            let starNode = cc.instantiate(this.starPre);
            let x = i * 40 - 100 + tt / 2;
           console.log(x, "x");
            starNode.setPosition(cc.v2(x, 0));
            this.nextStarNode.addChild(starNode);
        }
    }
}

/**
 *  绘制龙骨动画
 * @param skeletonName
 */
public createSkeleton(skeletonName) {

    let that = this;

    cc.resources.loadDir(skeletonName, null, function (err, asset) {

        // 检查资源加载
        if (err) {
            console.warn('预加载 :', " 失败,      原因:", err);
            return;
        }

        for (let k = 0; k < asset.length; ++k) {
            let res = asset[k];
            if (res instanceof dragonBones.DragonBonesAsset) {

                that.skeleton.dragonAsset = res;
            } else if (res instanceof dragonBones.DragonBonesAtlasAsset) {
                that.skeleton.dragonAtlasAsset = res;
            }
        }
```

```
            that.skeleton.armatureName = "armatureName";
            that.skeleton.playAnimation("Stand", 0);
        });
    }

    /**
     *  升星按钮
     */
    onClickStarUp() {
        let heroEny=heroData.ins.selectHeroEny;
        let chipNum = (heroData.ins.chipEntityDic.get(heroEny.heroId) &&
heroData.ins.chipEntityDic.get(heroEny.heroId).chipNum) || 0;

        if (chipNum<heroEny.upStarLvCost) {
            tipsTool.ins.createItemIconTips({ iconUrl: null, title: "您的碎片不足" });
            return;
        }
        else{
            heroData.ins.chipEntityDic.get(heroEny.heroId).chipNum-=heroEny.upStarLvCost;
            heroData.ins.upStar();
        }
    }

    /**
     *  升级   按钮
     */
    onClickLvUp() {
        let heroEny: HeroEntity = heroData.ins.selectHeroEny;
        console.log(heroData.ins.selectHeroEny)
        if (gameData.ins.gold < heroEny.upCost) {
            tipsTool.ins.createItemIconTips({ iconUrl: null, title: "您的金币不足" });
        } else {
            heroData.ins.upLv(heroEny.heroId);
        }
    }

    /**
    *  突破   按钮
    */
    onClickBreakUp() {
        SceneManager.getInstance().pushLayer("breakLayer");
    }

    // 注册按钮点击事件
    RegistBtnClickEvent(futcName: string, button: cc.Button) {
        var clickEventHandler = new cc.Component.EventHandler();
        clickEventHandler.target = this.node; //   这个  node  节点是你的事件处理代码组件所属的节点
        clickEventHandler.component = "heroDetailLayer";//  这个是代码文件名
        clickEventHandler.handler = futcName;
        // clickEventHandler.customEventData = btnLabel;
```

```
        button.clickEvents = [];//先清空
        button.clickEvents.push(clickEventHandler);
    }

    // 点击向右的箭头按钮
    onclickRightBtn() {
        let nextHeroId=-1;
        let keyList=heroData.ins.heroEntityDic.getKeyList();
        keyList.forEach((element,index) => {
          if (heroData.ins.selectHeroId==element) {
              nextHeroId=keyList[index+1];
          }
        });

        if (heroData.ins.heroEntityDic.containsKey(nextHeroId)) {
            heroData.ins.selectHeroId = nextHeroId;
        }
        else {
            console.log(" 超出 ")
          let list = heroData.ins.heroEntityDic.getValueList();
          heroData.ins.selectHeroId = list[0].heroId;
        }
        this.drawUI(heroData.ins.selectHeroEny);
    }



    // 点击向左的箭头按钮
    onclickLeftBtn() {
        let nextHeroId=-1;
        let keyList=heroData.ins.heroEntityDic.getKeyList();
        keyList.forEach((element,index) => {
          if (heroData.ins.selectHeroId==element) {
              nextHeroId=keyList[index-1];
          }
        });

        if (heroData.ins.heroEntityDic.containsKey(nextHeroId)) {
            heroData.ins.selectHeroId = nextHeroId;
        }
        else {
            console.log(" 超出 ")
          let list = heroData.ins.heroEntityDic.getValueList();
            heroData.ins.selectHeroId = list[list.length-1].heroId;
        }
        this.drawUI(heroData.ins.selectHeroEny);
    }

}
```

```
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
import { Dictionary } from "../../../libs/core/Dictionary";
import { drawScroll, drawScroll2 } from "../../../libs/core/utils";
import { SceneManager } from "../../../libs/manager/SceneManager";
import { tipsTool } from "../../../libs/tips/tipTool";
import gameData from "../../data/gameData";
import mailData from "../../data/mail/mailData";
import taskData from "../../data/task/taskData";
import { currency } from "../../enum/config";
import mainScene from "../mainScene";

const {ccclass, property} = cc._decorator;

@ccclass
export default class mailLayer extends BaseGameLayer {


    @property(cc.Node)
    viewContent: cc.Node = null;
    @property(cc.Prefab)
    mailComPre: cc.Prefab = null;

    onEnable(): void {
        super.onEnable();
        this.drawUI();

    }


    start() {

    }

    drawUI() {
        console.log(this.viewContent.children)
        this.viewContent.removeAllChildren();
        setTimeout(() => {
            drawScroll2(this.viewContent, mailData.ins.currExistMailArr, this.mailComPre, "mailCom", 1, 10,
50);
        }, 100);
    }

    // update (dt) {}

    // 点击删除所有领取完奖励的的邮件按钮
    onclickRemoveAllReceiveBtn() {
        if (mailData.ins.currExistMailArr.length<1) {
            tipsTool.ins.createItemIconTips({ iconUrl: null, title: "无删除邮件" });
            return;
        }
```

```
        let arr=[];
        let isHaveReceiveMail:boolean=true;
         mailData.ins.currExistMailArr.forEach(element => {
              if (element.isReceiveReward==false) {
                    arr.push(element);
              }
              else{isHaveReceiveMail=false}
         });
         if (isHaveReceiveMail == true) {
              tipsTool.ins.createItemIconTips({ iconUrl: null, title: "无删除邮件" });
              return;
         }

         if (!arr) {
               mailData.ins.currExistMailArr = [];
         }
         else {
              mailData.ins.currExistMailArr = arr;
         }

         mailData.ins.setLocal();

         // 清空原本列表
         this.viewContent.removeAllChildren();
         // 重新实例化列表
              drawScroll2(this.viewContent, mailData.ins.currExistMailArr, this.mailComPre, "mailCom", 1, 25,
25);
    }

    // 点击一键领取按钮
    onclickAllReceiveBtn(){
         if (mailData.ins.currExistMailArr.length<1) {
              tipsTool.ins.createItemIconTips({ iconUrl: null, title: "无领取邮件" });
              return;
         }
           // 弹出奖励窗口
          let rewardArr = [];

         let isNoHave:boolean=false;//有没有邮件满足领取条件
         // 逻辑
         mailData.ins.currExistMailArr.forEach(element => {
              if (element.isReceiveReward==false) {
                   if (element.rewardType==currency.gold) {
                        gameData.ins.gold+=element.rewardNum;
                        element.isReceiveReward=true;
                         isNoHave=true;

                         rewardArr.push({ icon: "texture/gold", name:" 金币 ", num: `${element.rewardNum}` })
                   }
                   else if (element.rewardType==currency.diamond) {
```

```
                            gameData.ins.diamond+=element.rewardNum;
                            element.isReceiveReward=true;
                             isNoHave=true;

                             rewardArr.push({ icon: "texture/diamond", name:" 钻石 ", num:
`${element.rewardNum}` })
                        }
                    }
                });
                if (isNoHave==false) {
                    tipsTool.ins.createItemIconTips({ iconUrl: null, title: "无领取邮件" });
                    return;
                }
                //UI
                this.viewContent.children.forEach(Element => {
                    Element.getChildByName("rewardImg").getChildByName("isReceiveImg").active = true;
                })

                gameData.ins.gameDataSetLocal();
                mailData.ins.setLocal();

                // 宝箱奖励
                SceneManager.getInstance().pushLayer("rewardLayer", rewardArr);
        }

        // 关闭按钮
        onclickCloseBtn(e) {
            super.onTouchUI(e, "close");
            SceneManager.getInstance().pushLayer("bottomLayer");
        }
}
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
import { Dictionary } from "../../../libs/core/Dictionary";
import { drawScroll, drawScroll2 } from "../../../libs/core/utils";
import { SceneManager } from "../../../libs/manager/SceneManager";
import { tipsTool } from "../../../libs/tips/tipTool";
import gameData from "../../data/gameData";
import mailData from "../../data/mail/mailData";
import taskData from "../../data/task/taskData";
import { currency } from "../../enum/config";
import mainScene from "../mainScene";

const {ccclass, property} = cc._decorator;

@ccclass
export default class mailLayer extends BaseGameLayer {


    @property(cc.Node)
    viewContent: cc.Node = null;
```

```
    @property(cc.Prefab)
     mailComPre: cc.Prefab = null;

    onEnable(): void {
           super.onEnable();
          this.drawUI();

    }


     start() {

    }

    drawUI() {
          console.log(this.viewContent.children)
          this.viewContent.removeAllChildren();
          setTimeout(() => {
               drawScroll2(this.viewContent, mailData.ins.currExistMailArr, this.mailComPre, "mailCom", 1, 10,
50);
           }, 100);
    }

    // update (dt) {}

    // 点击删除所有领取完奖励的的邮件按钮
    onclickRemoveAllReceiveBtn() {
          if (mailData.ins.currExistMailArr.length<1) {
               tipsTool.ins.createItemIconTips({ iconUrl: null, title: "无删除邮件" });
               return;
          }
          let arr=[];
          let isHaveReceiveMail:boolean=true;
           mailData.ins.currExistMailArr.forEach(element => {
               if (element.isReceiveReward==false) {
                    arr.push(element);
               }
               else{isHaveReceiveMail=false}
           });
           if (isHaveReceiveMail == true) {
               tipsTool.ins.createItemIconTips({ iconUrl: null, title: "无删除邮件" });
               return;
          }

          if (!arr) {
                mailData.ins.currExistMailArr = [];
          }
           else {
               mailData.ins.currExistMailArr = arr;
          }
```

```
            mailData.ins.setLocal();

            // 清空原本列表
            this.viewContent.removeAllChildren();
            // 重新实例化列表
                drawScroll2(this.viewContent, mailData.ins.currExistMailArr, this.mailComPre, "mailCom", 1, 25,
25);
        }

        // 点击一键领取按钮
        onclickAllReceiveBtn(){
            if (mailData.ins.currExistMailArr.length<1) {
                tipsTool.ins.createItemIconTips({ iconUrl: null, title: "无领取邮件" });
                return;
            }
              // 弹出奖励窗口
              let rewardArr = [];

            let isNoHave:boolean=false;//有没有邮件满足领取条件
            // 逻辑
            mailData.ins.currExistMailArr.forEach(element => {
                if (element.isReceiveReward==false) {
                    if (element.rewardType==currency.gold) {
                        gameData.ins.gold+=element.rewardNum;
                        element.isReceiveReward=true;
                        isNoHave=true;

                        rewardArr.push({ icon: "texture/gold", name:" 金币 ", num: `${element.rewardNum}` })
                    }
                    else if (element.rewardType==currency.diamond) {
                        gameData.ins.diamond+=element.rewardNum;
                        element.isReceiveReward=true;
                        isNoHave=true;

                        rewardArr.push({ icon: "texture/diamond", name:" 钻石 ", num:
`${element.rewardNum}` })
                    }
                }
            });
            if (isNoHave==false) {
                tipsTool.ins.createItemIconTips({ iconUrl: null, title: "无领取邮件" });
                return;
            }
            //UI
            this.viewContent.children.forEach(Element => {
                Element.getChildByName("rewardImg").getChildByName("isReceiveImg").active = true;
            })

            gameData.ins.gameDataSetLocal();
```

```
            mailData.ins.setLocal();

            // 宝箱奖励
             SceneManager.getInstance().pushLayer("rewardLayer", rewardArr);
        }

        // 关闭按钮
        onclickCloseBtn(e) {
            super.onTouchUI(e, "close");
            SceneManager.getInstance().pushLayer("bottomLayer");
        }
}
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
 import { GameFacade } from "../../../libs/core/GameFacade";
 import { PureMVCEvents } from "../../../libs/core/PureMVCEvents";
 import { randomNum } from "../../../libs/core/utils";
 import { SceneManager } from "../../../libs/manager/SceneManager";
import { tipsTool } from "../../../libs/tips/tipTool";
import gameData from "../../data/gameData";
 import heroData from "../../data/hero/heroData";
import taskData from "../../data/task/taskData";
import welfareDate from "../../data/welfare/welfareData";

 const { ccclass, property } = cc._decorator;

@ccclass
export default class recruitLayer extends BaseGameLayer {

        @property(cc.Label)
        diamondNumLabel: cc.Label = null;

        @property(cc.Button)
        ComfirmRecruitOneBtn: cc.Button = null;

        @property(cc.Button)
        ComfirmRecruitTenBtn: cc.Button = null;

        onEnable(): void {
             super.onEnable();
            this.drawUI();
        }


        start() {

        }

        drawUI(){
            // 更新拥有钻石数量 UI
            this.diamondNumLabel.string = gameData.ins.diamond.toString();
```

```
    }
    // update (dt) {}



    //  点击确认使用钻石招募一次英雄
    onclickComfirmRecruitOneBtn() {
        let diamondNumSpend = 1;
        if (gameData.ins.diamond >= diamondNumSpend) {
            gameData.ins.diamond -= diamondNumSpend;
            // 判断抽到的奖励类型
            let randowTypeNum = this.randowRewardType();
            switch (randowTypeNum) {
                case 1:// 武将
                    let id = this.getrandowSpareHeroId();
                    if (id == -1) {
                        console.debug("id 为负一")
                        return;
                    }
                    heroData.ins.getHero(id);
                    console.debug(` 得到英雄 id${id} 名字 ${heroData.ins.heroTpDic.get(id).name}`);

                    // 更新 UI
                    this.diamondNumLabel.string = gameData.ins.diamond.toString();
                    //
GameFacade.getInstance().sendNotification(PureMVCEvents.GetHERO_UI_UPDATE);

GameFacade.getInstance().sendNotification(PureMVCEvents.ON_HERO_EXCHANGE);


                    // 弹出奖励窗口
                    let rewardArr = [
                        { icon: "herolReward/1679282963779354", name:
`${heroData.ins.heroTpDic.get(id).name}`, num: 1 },
                    ]; // 宝箱奖励
                    SceneManager.getInstance().pushLayer("rewardLayer", rewardArr);
                    break;
                case 2:// 金币
                    let getGold: number = randomNum(100, 500);
                    gameData.ins.gold += getGold;
                    // 弹出奖励窗口
                    let rewardArr2 = [
                        { icon: "texture/gold", name: ` 金币 `, num: getGold },
                    ]; // 宝箱奖励
                    SceneManager.getInstance().pushLayer("rewardLayer", rewardArr2);
                    break;
                case 3:// 英雄碎片
                    let getChipNum = randomNum(1, 6);
                    let chipHeroArr: any;
                    chipHeroArr = heroData.ins.heroTpDic.getKeyList().concat([]);
```

```
                    let randowHeroId = chipHeroArr[randomNum(0, chipHeroArr.length-1)];
                    let chipEn=heroData.ins.chipEntityDic.get(randowHeroId);
                    chipEn.chipNum += getChipNum;
                    // 弹出奖励窗口
                    // icon: chipEn.chipUrl
                    let rewardArr3 = [
                        { icon: "texture/16793892815371", name: chipEn.chipName, num: getChipNum },
                    ]; // 宝箱奖励
                    SceneManager.getInstance().pushLayer("rewardLayer", rewardArr3);
                    break;
            }
            // 进行过招募，增加每日任务招募次数
            taskData.ins.dailyRecruitMun += 1;

            welfareDate.ins.recruitNumTotal+=1;

            // 储存
            gameData.ins.gameDataSetLocal();
            welfareDate.ins.welfareSetLocal();
            heroData.ins.setLocal();
            taskData.ins.taskSetLocal();
        }
        else {
            tipsTool.ins.createItemIconTips({ iconUr: null, title: "钻石不足" });
        }
    }

    //  点击确认使用钻石招募十次英雄
    onclickComfirmRecruitTenBtn() {
    //      let diamondNumSpend = 10;
    //      if (gameData.ins.diamond >= diamondNumSpend) {
    //          gameData.ins.diamond -= diamondNumSpend;
    //          for (let i = 0; i < 10; i++) {

    //          }
    //      }
    //      else {
    //              console.debug(" 钻石不足 ");
    //              tipsTool.ins.createItemIconTips({ iconUrl: null, title: "钻石不足" });
    //          }

        }

    // 随机给个玩家还未拥有的英雄id
    getrandowSpareHeroId() {
        let spaceHeroArr:any;
        spaceHeroArr=heroData.ins.heroTpDic.getKeyList().concat([]);
        heroData.ins.heroEntityDic.getKeyList().forEach(key => {
            for (let t = 0; t < spaceHeroArr.length; t++) {
                if (spaceHeroArr[t] == key) {
```

```
                        spaceHeroArr.splice(t, 1);
                    }
                }
            });

            let randowHeroId = spaceHeroArr[Math.round(Math.random() * spaceHeroArr.length - 1)]
            if (!randowHeroId) {
                randowHeroId = -1;
            }
            return randowHeroId;
        }

        // 需要抽奖时，随机给个奖励 ( 英雄，金币，英雄碎片 )
        randowRewardType():number{
            let ranNum = randomNum(1, 3);
            if (ranNum == 1) {// 武将
                return 1;
            }
            else if (ranNum == 2) {//金币
                return 2;
            }
            else if (ranNum == 3) {//英雄碎片
                return 3;
            }
        }

        // 关闭按钮
        onclickCloseBtn(e) {
            super.onTouchUI(e, "close");
            SceneManager.getInstance().pushLayer("bottomLayer");
        }

        // 点击充值按钮
        onclickRechargeBtn(){
            SceneManager.getInstance().pushLayer("rechargeLayer");
            SceneManager.getInstance().removeLayer("recruitLayer");
            SceneManager.getInstance().removeLayer("bottomLayer");
        }

}
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
import { Dictionary } from "../../../libs/core/Dictionary";
import { drawScroll, drawScroll2 } from "../../../libs/core/utils";
import { SceneManager } from "../../../libs/manager/SceneManager";
import { tipsTool } from "../../../libs/tips/tipTool";
import gameData from "../../data/gameData";
import mailData from "../../data/mail/mailData";
import setData from "../../data/set/setData";
import taskData from "../../data/task/taskData";
import { currency } from "../../enum/config";
```

```
import mainScene from "../mainScene";

const {ccclass, property} = cc._decorator;

@ccclass
 export default class setLayer extends BaseGameLayer {


        @property(cc.Node)
        effectsOption:cc.Node=null;

        @property(cc.Node)
         musicOption:cc.Node=null;

        onEnable(): void {
               super.onEnable();
              this.drawUI();
        }


         start() {

        }

        drawUI() {

        }

        // update (dt) {}


        // 关闭按钮
        onclickCloseBtn(e) {
              super.onTouchUI(e, "close");
              SceneManager.getInstance().pushLayer("bottomLayer");
        }

       onclickUserArgeementLabel(){
               SceneManager.getInstance().removeLayer("setLayer");
             SceneManager.getInstance().pushLayer("userAgreementLayer");
        }

       // 点击音效开关按钮
        onclickEffectBtn() {
              this.effectsOption.getComponent(cc.Button).enabled=false;// 不能连续点击按钮多次
              if (setData.ins.IsEffectOpen) {
                    this.effectsOption.getChildByName("Label").active = false;

                    cc.tween(this.effectsOption.getChildByName("icon"))
                          .to(0.2, { x: 35 }, cc.easeInOut(3))
```

```
                .call(() => {
                    cc.resources.load("texture/set/close-bg", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
                        this.effectsOption.getChildByName("bg").getComponent(cc.Sprite).spriteFrame =
res;
                    })
                     this.effectsOption.getChildByName("Label2").active = true;
                })
                .start()

            setData.ins.IsEffectOpen = false;

        }
        else {
            this.effectsOption.getChildByName("Label2").active = false;

            cc.tween(this.effectsOption.getChildByName("icon"))
                 .to(0.2, { x: -34 }, cc.easeInOut(3))
                .call(() => {
                    cc.resources.load("texture/set/open-bg", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
                        this.effectsOption.getChildByName("bg").getComponent(cc.Sprite).spriteFrame =
res;
                    })
                    this.effectsOption.getChildByName("Label").active = true;
                })
                .start()

            setData.ins.IsEffectOpen = true;
        }

        setTimeout(() => {// 不能连续点击按钮多次
            this.effectsOption.getComponent(cc.Button).enabled=true;
            }, 500);

    }

    // 点击音乐开关按钮
    onclickMusicBtn(){
        this.musicOption.getComponent(cc.Button).enabled=false;// 不能连续点击按钮多次
        if (setData.ins.IsMusic) {
            this.musicOption.getChildByName("Label").active = false;

            cc.tween(this.musicOption.getChildByName("icon"))
                .to(0.2, { x: 35 }, cc.easeInOut(3))
                .call(() => {
                    cc.resources.load("texture/set/close-bg", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
                        this.musicOption.getChildByName("bg").getComponent(cc.Sprite).spriteFrame =
res;
                    })
                    this.musicOption.getChildByName("Label2").active = true;
                })
```

```
                    .start()

                setData.ins.IsMusic = false;

            }
            else {
                this.musicOption.getChildByName("Label2").active = false;

                cc.tween(this.musicOption.getChildByName("icon"))
                    .to(0.2, { x: -34 }, cc.easeInOut(3))
                    .call(() => {
                        cc.resources.load("texture/set/open-bg", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
                            this.musicOption.getChildByName("bg").getComponent(cc.Sprite).spriteFrame =
res;
                        })
                        this.musicOption.getChildByName("Label").active = true;
                    })
                    .start()

                setData.ins.IsMusic = true;
            }

            setTimeout(() => {// 不能连续点击按钮多次
                this.musicOption.getComponent(cc.Button).enabled = true;
            }, 500);

        }

        // 点击礼包码按钮
        onclickGoGiftCodeBtn() {
            SceneManager.getInstance().pushLayer("giftCodeLayer");
        }

}
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
import { DateUtil } from "../../../libs/core/DateUtil";
import { Dictionary } from "../../../libs/core/Dictionary";
import { drawScroll, drawScroll2 } from "../../../libs/core/utils";
import { LocalStorage } from "../../../libs/manager/LocalStorage";
import { SceneManager } from "../../../libs/manager/SceneManager";
import gameData from "../../data/gameData";
import taskData from "../../data/task/taskData";
import { currency } from "../../enum/config";
import mainScene from "../mainScene";

const {ccclass, property} = cc._decorator;

@ccclass
export default class taskLayer extends BaseGameLayer {
```

```
// 拖入
 @property(cc.Node)
tab1ViewContent: cc.Node = null;
 @property(cc.Node)
tab2ViewContent: cc.Node = null;
@property(cc.Prefab)
 taskAchieveComPre: cc.Prefab = null;
@property(cc.Prefab)
taskActionComPre: cc.Prefab = null;
 @property(cc.Node)
tab2:cc.Node=null;
 @property(cc.Node)
 tab1: cc.Node = null;
 @property(cc.Node)
weekTaskPlate: cc.Node = null;
 @property(cc.Node)
dailyTaskPlate: cc.Node = null;
@property(cc.Sprite)
 option1Background:cc.Sprite=null;
@property(cc.Sprite)
 option2Background:cc.Sprite=null;


static ins: taskLayer = null;

 onEnable(): void {
        super.onEnable();
       this.drawUITab1();

        if (taskLayer.ins==null) {
            taskLayer.ins=this;
        }
}


 start() {

}

 drawUITab2() {// 成就
      taskData.ins.updateAchieveTaskItemArrByProgress();// 打开时更新一下任务列表数组
      setTimeout(() => {
          drawScroll2(this.tab2ViewContent, taskData.ins.taskAchieveItemArr, this.taskAchieveComPre,
"taskAchieveCom", 1, 10, 10);
       }, 100);
}

 drawUITab1() {// 每日
      // 绘制每周任务每日任务奖励栏
      this.drawTaskPlateUI();
```

```
        // 任务列表要进行 getActionTaskItem 进行筛选得到真正的任务列表
        setTimeout(() => {
            drawScroll2(this.tab1ViewContent, taskData.ins.getActionTaskItem(), this.taskActionComPre,
"taskActionCom", 1, 10, 10);
        }, 100);
    }

    // 绘制每周活动领取栏
    drawTaskPlateUI() {
        // 周活跃奖励栏
        this.weekTaskPlate.getChildByName("ActiveNum").getComponent(cc.Label).string =
taskData.ins.WeekActionNum.toString();
        this.weekTaskPlate.getChildByName("scorePro").getComponent(cc.ProgressBar).progress =
taskData.ins.WeekActionNum/500;

        cc.resources.load("configs/weekTaskPlateReward", cc.JsonAsset, (err, res: cc.JsonAsset) => {
            res.json.forEach((element,index) => {
                if (element.rewardType==currency.gold) {
                    cc.resources.load("texture/gold",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{

this.weekTaskPlate.getChildByName(`RewardItem${index}`).getChildByName("iconBg").getComponent(cc.Spr
ite).spriteFrame=res;
                    })
                    cc.resources.load("texture/task/green-frame",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{

this.weekTaskPlate.getChildByName(`RewardItem${index}`).getChildByName("iconSquare").getComponent(cc.
Sprite).spriteFrame=res;
                    })
                }
                else if(element.rewardType==currency.diamond){
                    cc.resources.load("texture/diamond",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{

this.weekTaskPlate.getChildByName(`RewardItem${index}`).getChildByName("iconBg").getComponent(cc.Spr
ite).spriteFrame=res;
                    })
                    cc.resources.load("texture/task/purple-frame", cc.SpriteFrame, (err, res: cc.SpriteFrame)
=> {

this.weekTaskPlate.getChildByName(`RewardItem${index}`).getChildByName("iconSquare").getComponent(cc.
Sprite).spriteFrame = res;
                    })
                }


this.weekTaskPlate.getChildByName(`RewardItem${index}`).getChildByName("rewardNum").getComponent(c
c.Label).string = element.rewardNum;
            });
        })

        //#region  显示满足条件的奖励样式，和已经领取过的样式
```

```
        if (taskData.ins.alreadyReceiveReward2Arr[0] && taskData.ins.alreadyReceiveReward2Arr[0].length >
0) {
            taskData.ins.alreadyReceiveReward2Arr[0].forEach((element) => {

this.weekTaskPlate.getChildByName(`RewardItem${element}`).getChildByName("cancelMark").active = true;
            })
        }

        let WeekNum = taskData.ins.WeekActionNum;
        if (50 <= WeekNum && WeekNum < 150) {
            this.weekTaskPlate.getChildByName(`RewardItem${0}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
        }
        else if (150 <= WeekNum && WeekNum < 250) {
            this.weekTaskPlate.getChildByName(`RewardItem${0}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
            this.weekTaskPlate.getChildByName(`RewardItem${1}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
        }
        else if (250 <= WeekNum && WeekNum < 350) {
            this.weekTaskPlate.getChildByName(`RewardItem${0}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
            this.weekTaskPlate.getChildByName(`RewardItem${1}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
            this.weekTaskPlate.getChildByName(`RewardItem${2}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
        }
        else if (350 <= WeekNum && WeekNum < 450) {
            this.weekTaskPlate.getChildByName(`RewardItem${0}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
            this.weekTaskPlate.getChildByName(`RewardItem${1}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
            this.weekTaskPlate.getChildByName(`RewardItem${2}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
            this.weekTaskPlate.getChildByName(`RewardItem${3}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
        }
        else if (450 <= WeekNum) {
            this.weekTaskPlate.getChildByName(`RewardItem${0}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
            this.weekTaskPlate.getChildByName(`RewardItem${1}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
            this.weekTaskPlate.getChildByName(`RewardItem${2}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
            this.weekTaskPlate.getChildByName(`RewardItem${3}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
            this.weekTaskPlate.getChildByName(`RewardItem${4}`).getChildByName("iconSquare").color =
cc.Color.WHITE;
        }
        //#endregion
```

```
            // 更新奖励栏上领取奖励初事件
             for (let i = 0; i < 5; i++) {
                  if
(this.weekTaskPlate.getChildByName(`RewardItem${i}`).getChildByName("iconSquare").color.equals(cc.Color.
WHITE)==true&&this.weekTaskPlate.getChildByName(`RewardItem${i}`).getChildByName("cancelMark").act
ive==false) {
                        this.RegistBtnClickEvent(true, i);
                  }
            }

            // 日活跃奖励栏
            this.dailyTaskPlate.getChildByName("ActiveNum").getComponent(cc.Label).string =
taskData.ins.DailyActionNum.toString();
             console.log(taskData.ins.DailyActionNum)
             this.dailyTaskPlate.getChildByName("scorePro").getComponent(cc.ProgressBar).progress =
taskData.ins.DailyActionNum / 200;


            cc.resources.load("configs/dailyTaskPlateReward", cc.JsonAsset, (err, res: cc.JsonAsset) => {
                  res.json.forEach((element, index) => {
                        if (element.rewardType ==currency.gold) {
                              cc.resources.load("texture/gold",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{

 this.dailyTaskPlate.getChildByName(`RewardItem${index}`).getChildByName("iconBg").getComponent(cc.Spr
ite).spriteFrame=res;
                              })
                              cc.resources.load("texture/task/green-frame",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{

 this.dailyTaskPlate.getChildByName(`RewardItem${index}`).getChildByName("iconSquare").getComponent(cc.
Sprite).spriteFrame=res;
                              })
                        }
                         else if(element.rewardType==currency.diamond){
                               cc.resources.load("texture/diamond",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{

 this.dailyTaskPlate.getChildByName(`RewardItem${index}`).getChildByName("iconBg").getComponent(cc.Spr
ite).spriteFrame=res;
                              })
                              cc.resources.load("texture/task/purple-frame",cc.SpriteFrame,(err,res:cc.SpriteFrame)=>{

 this.dailyTaskPlate.getChildByName(`RewardItem${index}`).getChildByName("iconSquare").getComponent(cc.
Sprite).spriteFrame=res;
                              })
                        }


 this.dailyTaskPlate.getChildByName(`RewardItem${index}`).getChildByName("rewardNum").getComponent(c
c.Label).string=element.rewardNum;
                  });
```

```
        })


         //#region  显示满足条件的奖励样式，和已经领取过的样式
         if (taskData.ins.alreadyReceiveReward2Arr[1]&&taskData.ins.alreadyReceiveReward2Arr[1].length>0)
{
              taskData.ins.alreadyReceiveReward2Arr[1].forEach((element)=>{

 this.dailyTaskPlate.getChildByName(`RewardItem${element}`).getChildByName("cancelMark").active=true;
              })
         }

         let dailyNum=taskData.ins.DailyActionNum;
        if (20 <= dailyNum && dailyNum < 60) {

 this.dailyTaskPlate.getChildByName(`RewardItem${0}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;
         }
         else if (60 <= dailyNum && dailyNum < 100) {

 this.dailyTaskPlate.getChildByName(`RewardItem${0}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;

 this.dailyTaskPlate.getChildByName(`RewardItem${1}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;
         }
        else if (100 <= dailyNum && dailyNum < 140) {

 this.dailyTaskPlate.getChildByName(`RewardItem${0}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;

 this.dailyTaskPlate.getChildByName(`RewardItem${1}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;

 this.dailyTaskPlate.getChildByName(`RewardItem${2}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;
         }
        else if (140 <= dailyNum && dailyNum < 180) {

 this.dailyTaskPlate.getChildByName(`RewardItem${0}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;

 this.dailyTaskPlate.getChildByName(`RewardItem${1}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;

 this.dailyTaskPlate.getChildByName(`RewardItem${2}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;

 this.dailyTaskPlate.getChildByName(`RewardItem${3}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;
```

```
            }
            else if (180 <= dailyNum) {

this.dailyTaskPlate.getChildByName(`RewardItem${0}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;

this.dailyTaskPlate.getChildByName(`RewardItem${1}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;

this.dailyTaskPlate.getChildByName(`RewardItem${2}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;

this.dailyTaskPlate.getChildByName(`RewardItem${3}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;

this.dailyTaskPlate.getChildByName(`RewardItem${4}`).getChildByName("iconSquare").color=cc.Color.WHIT
E;
            }
            //#endregion

              // 更新奖励栏上领取奖励初事件
            for (let i = 0; i < 5; i++) {
                if
(this.dailyTaskPlate.getChildByName(`RewardItem${i}`).getChildByName("iconSquare").color.equals(cc.Color.
WHITE)==true&&this.dailyTaskPlate.getChildByName(`RewardItem${i}`).getChildByName("cancelMark").act
ive==false) {
                    this.RegistBtnClickEvent(false, i);
                }
            }

        }

        // update (dt) {}

        // 关闭按钮
        onclickCloseBtn(e) {
            super.onTouchUI(e, "close");
            SceneManager.getInstance().pushLayer("bottomLayer");

        }

        // 点击每日任务
        onclickOption1Btn() {
            this.tab1ViewContent.removeAllChildren();
            this.tab2.active = false;
            this.tab1.active=true;

            this.drawUITab1();

            // 更换图片
```

```
            cc.resources.load("texture/task/highlightTabBtn", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
                this.option1Background.spriteFrame = res;
            })
            cc.resources.load("texture/task/lowLightTabBtn", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
                this.option2Background.spriteFrame = res;
            })
        }
        // 点击成就
        onclickOption2Btn() {
            this.tab2ViewContent.removeAllChildren();
            this.tab2.active = true;
            this.tab1.active = false;

            this.drawUITab2();

            // 更换图片
            cc.resources.load("texture/task/highlightTabBtn", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
                this.option2Background.spriteFrame = res
            })
            cc.resources.load("texture/task/lowLightTabBtn", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
                this.option1Background.spriteFrame = res
            })
        }

        // 注册奖励栏点击事件，获取奖励
        RegistBtnClickEvent(isWeekBar: boolean, btnInde) {//btnindex 为第几个 btn
            var clickEventHandler = new cc.Component.EventHandler();
            clickEventHandler.target = this.node; //  这个  node  节点是你的事件处理代码组件所属的节点
            clickEventHandler.component = "taskLayer";//  这个是代码文件名
            clickEventHandler.customEventData = isWeekBar + "," + btnInde;
            clickEventHandler.handler = "onclickRewardBarBtn";

            if (isWeekBar) {
                var button =
this.weekTaskPlate.getChildByName(`RewardItem${btnInde}`).getComponent(cc.Button);
            }
            else {
                var button =
this.dailyTaskPlate.getChildByName(`RewardItem${btnInde}`).getComponent(cc.Button);
            }
            button.clickEvents=[];// 先清空
            button.clickEvents.push(clickEventHandler);
        }

        // 奖励栏奖励点击事件
        onclickRewardBarBtn(event, twoParameter) {
            let arr = twoParameter.split(",")// 包含两个输入参数的数组
            let isWeekBar = arr[0];
            let btnIndex = arr[1];
            if (isWeekBar == "true") {
```

```
                // 显示已领取样式

this.weekTaskPlate.getChildByName(`RewardItem${btnIndex}`).getChildByName("cancelMark").active = true;

                // 弹出奖励窗口
                cc.resources.load("configs/weekTaskPlateReward", cc.JsonAsset, (err, res: cc.JsonAsset) => {
                    let newJson = res.json;
                    let rewardName = taskData.ins.getNamestrByCurrency(newJson[btnIndex].rewardType);
                    console.log(rewardName)
                    let Num = newJson[btnIndex].rewardNum;
                    let path = taskData.ins.getPathstrByCurrency(newJson[btnIndex].rewardType);
                    let rewardArr = [
                        { icon: `${path}`, name: rewardName, num: Num },
                    ]; // 宝箱奖励
                    SceneManager.getInstance().pushLayer("rewardLayer", rewardArr);

                    //UI
                    if (newJson[btnIndex].rewardType == currency.gold) {
                        gameData.ins.gold += newJson[btnIndex].rewardNum;
                    }
                    else if (newJson[btnIndex].rewardType == currency.diamond) {
                        gameData.ins.diamond += newJson[btnIndex].rewardNum;
                    }

                    gameData.ins.gameDataSetLocal();
                })
                console.log(taskData.ins.alreadyReceiveReward2Arr[0])
                taskData.ins.alreadyReceiveReward2Arr[0].push(btnIndex);

                // 保存
                taskData.ins.taskSetLocal();
                // 使用后注销按钮点击事件

this.weekTaskPlate.getChildByName(`RewardItem${btnIndex}`).getComponent(cc.Button).clickEvents=[];
            }
            else if(isWeekBar=="false"){
                // 显示已领取样式

this.dailyTaskPlate.getChildByName(`RewardItem${btnIndex}`).getChildByName("cancelMark").active=true;

                // 弹出奖励窗口
                cc.resources.load("configs/dailyTaskPlateReward", cc.JsonAsset, (err, res: cc.JsonAsset) => {
                    let newJson = res.json;
                    let rewardName = taskData.ins.getNamestrByCurrency(newJson[btnIndex].rewardType);
                    console.log(rewardName)
                    let Num = newJson[btnIndex].rewardNum;
                    let path=taskData.ins.getPathstrByCurrency(newJson[btnIndex].rewardType);
                    let rewardArr = [
                        { icon: `${path}`, name: rewardName, num: Num },
                    ]; // 宝箱奖励
```

```
                    SceneManager.getInstance().pushLayer("rewardLayer", rewardArr);

                    //UI
                    if (newJson[btnIndex].rewardType == currency.gold) {
                        gameData.ins.gold += newJson[btnIndex].rewardNum;
                    }
                    else if (newJson[taskData.ins.dailyRewardReceiveWhere].rewardType == currency.diamond)
{
                        gameData.ins.diamond += newJson[btnIndex].rewardNum;
                    }

                    gameData.ins.gameDataSetLocal();
                })
                taskData.ins.alreadyReceiveReward2Arr[1].push(btnIndex);

                // 保存
                taskData.ins.taskSetLocal();

                // 使用后注销按钮点击事件

this.dailyTaskPlate.getChildByName(`RewardItem${btnIndex}`).getComponent(cc.Button).clickEvents = [];
            }

        }


        test() {
            taskData.ins.WeekActionNum += 50;
            taskData.ins.DailyActionNum += 20;
            console.debug(taskData.ins.WeekActionNum)

            // taskData.ins.dailyRecruitMun+=1;
        }
}
import BaseGameLayer from "../../../libs/core/BaseGameLayer";
import { drawScroll2 } from "../../../libs/core/utils";
import { SceneManager } from "../../../libs/manager/SceneManager";
import welfareDate from "../../data/welfare/welfareData";

const {ccclass, property} = cc._decorator;

@ccclass
export default class welfareLayer extends BaseGameLayer {

    @property(cc.Node)
    tab1ScrollContent: cc.Node=null;
    @property(cc.Node)
    tab2ScrollContent: cc.Node=null;
    @property(cc.Node)
    tab3ScrollContent: cc.Node=null;
```

```
    @property(cc.Node)
    tab4ScrollContent: cc.Node=null;
    @property(cc.Prefab)
    welfareComPre:cc.Prefab=null;
    @property(cc.Prefab)
    welfareComPre2:cc.Prefab=null;
    @property(cc.Prefab)
    welfareComPre3:cc.Prefab=null;
    @property(cc.Prefab)
    welfareComPre4:cc.Prefab=null;
    @property(cc.Node)
    tab1:cc.Node=null;
    @property(cc.Node)
    tab2:cc.Node=null;
    @property(cc.Node)
    tab3:cc.Node=null;
    @property(cc.Node)
    tab4:cc.Node=null;
    @property(cc.Sprite)
    option1Background:cc.Sprite=null;
    @property(cc.Sprite)
    option2Background:cc.Sprite=null;
    @property(cc.Sprite)
    option3Background:cc.Sprite=null;
    @property(cc.Sprite)
    option4Background:cc.Sprite=null;


    static ins:welfareLayer=null;

    onEnable(): void {
        super.onEnable();
        this.drawUITab1();

        if (welfareLayer.ins==null) {
            welfareLayer.ins=this;
        }
    }


    start() {

    }

    drawUITab1() {
        this.tab1ScrollContent.removeAllChildren();
        setTimeout(() => {
            drawScroll2(this.tab1ScrollContent, welfareDate.ins.welfareGuanYuItemArr, this.welfareComPre,
"welfareCom", 1, 10, 20);
        }, 100);
```

```
    }

    drawUITab2(){
        this.tab2ScrollContent.removeAllChildren();
        setTimeout(() => {
            drawScroll2(this.tab2ScrollContent, welfareDate.ins.welfareDiaochanItemArr,
this.welfareComPre2, "welfareCom2", 1, 10, 20);
        }, 100);
    }

    drawUITab3(){
        this.tab3ScrollContent.removeAllChildren();
        setTimeout(() => {
            drawScroll2(this.tab3ScrollContent, welfareDate.ins.welfareZhaomudabiaoItemArr,
this.welfareComPre3, "welfareCom3", 1, 10, 20);
        }, 100);
    }

    drawUITab4(){
        this.tab4ScrollContent.removeAllChildren();
        setTimeout(() => {
            drawScroll2(this.tab4ScrollContent, welfareDate.ins.welfareZhaomushangchengItemArr,
this.welfareComPre4, "welfareCom4", 1, 10, 20);
        }, 100);
    }

    // update (dt) {}

    // 关闭按钮
    onclickCloseBtn(e) {
        super.onTouchUI(e, "close");
        SceneManager.getInstance().pushLayer("bottomLayer");
    }

    // 点击武圣关羽
    onclickOption1Btn() {
        this.tab4.active=false;
        this.tab3.active=false;
        this.tab2.active = false;
        this.tab1.active = true;

        this.drawUITab1();


        //#region  更换图片
        cc.resources.load("texture/welfare/guanyu-selected", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option1Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/diaochan-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
```

```
            this.option2Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/recruit-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option3Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/shop-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option4Background.spriteFrame = res;
        })
        //#endregion
    }
    // 点击五星貂蝉
    onclickOption2Btn() {
        this.tab4.active=false;
        this.tab3.active=false;
        this.tab2.active = true;
        this.tab1.active = false;

        this.drawUITab2();


        //#region  更换图片
        cc.resources.load("texture/welfare/guanyu-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option1Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/diaochan-selected", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option2Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/recruit-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option3Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/shop-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option4Background.spriteFrame = res;
        })
        //#endregion

    }

    // 点击招募达标
    onclickOption3Btn() {
        this.tab4.active=false;
        this.tab3.active=true;
        this.tab2.active = false;
        this.tab1.active = false;

        this.drawUITab3();


        //#region  更换图片
        cc.resources.load("texture/welfare/guanyu-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option1Background.spriteFrame = res;
```

```
        })
        cc.resources.load("texture/welfare/diaochan-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option2Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/recruit-selected", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option3Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/shop-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option4Background.spriteFrame = res;
        })
        //#endregion

    }

    // 点击招募商城
    onclickOption4Btn() {
        this.tab4.active=true;
        this.tab3.active=false;
        this.tab2.active = false;
        this.tab1.active = false;

        this.drawUITab4();


        //#region  更换图片
        cc.resources.load("texture/welfare/guanyu-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option1Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/diaochan-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option2Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/recruit-default", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option3Background.spriteFrame = res;
        })
        cc.resources.load("texture/welfare/shop-selected", cc.SpriteFrame, (err, res: cc.SpriteFrame) => {
            this.option4Background.spriteFrame = res;
        })
        //#endregion

    }
```