var UM:UserManager,TM:TimeManager,EM:EventManager ,CM:CacheManager,DM:DebugManager

class Main extends eui.UILayer {

/\*\*

\* 加载进度界面

\* loading process interface

\*/

private loadingView: MainLoadingUI;

protected createChildren(): void {

super.createChildren();

console.log('\_10')

//inject the custom material parser

//注入自定义的素材解析器

var assetAdapter = new AssetAdapter();

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

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

this.setScaleMode();

//设置加载进度界面

this.loadingView = MainLoadingUI.getInstance();

// initialize the Resource loading library

//初始化Resource资源加载库

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

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

UM = UserManager.getInstance();

TM = TimeManager.getInstance();

EM = EventManager.getInstance();

CM = CacheManager.getInstance();

DM = DebugManager.getInstance();

Config.initURLRequest();

console.log('\_1a')

}

private setScaleMode(){

}

/\*\*

\* 配置文件加载完成,开始预加载皮肤主题资源和preload资源组。

\* Loading of configuration file is complete, start to pre-load the theme configuration file and the preload resource group

\*/

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

console.log('\_1b')

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

// load skin theme configuration file, you can manually modify the file. And replace the default skin.

//加载皮肤主题配置文件,可以手动修改这个文件。替换默认皮肤。

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

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

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

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

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

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

RES.loadGroup("preload\_png");

}

private isThemeLoadEnd: boolean = false;

/\*\*

\* 主题文件加载完成,开始预加载

\* Loading of theme configuration file is complete, start to pre-load the

\*/

private onThemeLoadComplete(): void {

this.isThemeLoadEnd = true;

console.log('\_1c')

this.createScene();

}

private isResourceLoadEnd: boolean = false;

/\*\*

\* preload资源组加载完成

\* preload resource group is loaded

\*/

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

console.log('\_1d')

if (event.groupName == "preload\_png") {

this.isResourceLoadEnd = true;

this.removeLoadEvent();

this.createScene();

}

}

private removeLoadEvent(){

this.loadingView.setFinish();

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

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

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

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

}

private createScene(){

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

this.startCreateScene();

}

}

/\*\*

\* 资源组加载出错

\* The resource group loading failed

\*/

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

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

}

/\*\*

\* 资源组加载出错

\* Resource group loading failed

\*/

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

//TODO

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

//忽略加载失败的项目

//ignore loading failed projects

this.onResourceLoadComplete(event);

}

/\*\*

\* preload资源组加载进度

\* loading process of preload resource

\*/

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

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

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

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

}

}

/\*\*

\* 创建场景界面

\* Create scene interface

\*/

protected startCreateScene(): void {

CM.initData(RES.getRes("data\_json"));

GameManager.stage = this.stage;

GameManager.container = this;

if(App.isIOS){

GameManager.stage.frameRate = 60;

}

GameManager.getInstance().init();

console.log('\_11')

if(\_get['hide'])

return;

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

GameUI.getInstance().show();

});

}

}

class PKBuffData {

public user; //发起者,对象

public owner; //拥有者

public isDebuff = false;

public removeAble = true;

public endTime = 0;//到期时间 为0则为图腾类效果

public add = {}; //改变的值

public state = {}; //改变的状态

public tempValue = {};

public haveState = false

public haveValue = false

//以下为唯一技能才用到

public id;//唯一ID

public ing//这BUFF是否有起作用

public value;//技能等级数值,用于比较相同ID下哪个BUFF强

public endFun

constructor(obj?) {

}

//def,atk,addSpeed,hpChange

public addValue(key,value){

this.add[key] = value

this.haveValue = true

}

public addState(key){

this.state[key] = true

this.haveState = true

}

//使技能生效

public enable(){

if(this.ing)

return;

this.ing = true;

for(var s in this.add)

this.owner[s] += this.add[s];

}

//使技能无效

public disable(){

if(!this.ing)

return;

this.ing = false;

for(var s in this.add)

this.owner[s] -= this.add[s];

}

public remove(){

this.disable();

this.endFun && this.endFun(this);

}

}

class PKData extends egret.EventDispatcher{

private static instance:PKData;

private static instance2:PKData;

public static instanceIndex = 1

public static getInstance():PKData {

if(this.instanceIndex == 1)

{

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

return this.instance;

}

else

{

if (!this.instance2) this.instance2 = new PKData();

return this.instance2;

}

}

public currentState = 'def'

public quick = false//快速算出结果

public quickTime = Number.MAX\_VALUE//快速算出到这个时间

public baseData//原始PK数据

public isReplay;

public replayEndTime;

public isAuto;

public round; //当前的回合

public jumpMV = false;

public isGameOver = false //游戏结束

public showTopNum = 0 //头顶显示敌人出怪的数量

public startTime = 0 //游戏开始时间

public stopTime = 0 //游戏暂停时间

public actionTime = 0 //游戏数据上处理过的时间

public beginAuto = false //正式开始前的上怪处理

public monsterID;//怪物ID下标累计

public team1:PKTeamData; //进攻方

public team2:PKTeamData;

public playerNum = 2;

public endless = 0;//无尽时的倒计时

public needcd = 0;//限时的倒计时

public monsterChange = false//怪有变化

public randomSeed = 0//随机的种子

public randomTimes = 0//随机的次数

public monsterList = [];//场上的怪的数据

public playerObj = {};//场上的玩家的数据

public myPlayer:PKPlayerData;

public diamondData;

public heroStep;

public preLoadHeroStep;

public history = {};

public disableKey = {}; //同一时间不能起效的KEY

public actionRecord = []

public playSpeed = 1;//播放速度

public lastDealSpeedTime = 0;//当前播放速度开始时间

public speedAddTime = 0;//当前播放速度开始时间

constructor(){

super();

}

public isView(){

return this.isAuto || this.isReplay

}

public canSpeed(){

return false//this.isView() || (!PKManager.getInstance().isOnline && !GuideManager.getInstance().isGuiding)

}

//public changeSpeed(speed){

// this.playSpeed = speed;

//}

//取经过的时间

public getPassTime(){

var t = TM.nowMS();

var cd = t - (this.lastDealSpeedTime || this.startTime);

if(cd)

{

this.speedAddTime += (this.playSpeed - 1)\*cd;

}

this.lastDealSpeedTime = t;

return TM.nowMS() - this.startTime + this.speedAddTime;

}

//暂停

public stop(){

this.stopTime = TM.nowMS();

}

//继续

public play(){

if(this.stopTime)

{

var add = (TM.nowMS() - this.stopTime)\*this.playSpeed;

this.startTime += add

//this.actionTime += add;

this.stopTime = 0;

}

}

//初始化游戏

public init(data){

this.startTime = 0;

this.round = 1;

this.heroStep = 0;

this.preLoadHeroStep = 0;

this.isReplay = false;

this.isAuto = false;

this.baseData = data;

this.actionRecord = [];

this.quick = false

this.quickTime = Number.MAX\_VALUE

this.history = {};

this.monsterList.length = 0;

this.playerObj = {};

this.myPlayer = null;

this.actionTime = 0;

this.playSpeed = 1;

this.lastDealSpeedTime = 0;

this.speedAddTime = 0;

this.stopTime = 0;

this.monsterID = 1;

this.isGameOver = false;

this.monsterChange = false;

this.beginAuto = false;

this.currentState = 'pk';

PKMonsterAction.getInstance().init()

this.randomTimes = 0;

this.randomSeed = data.seed;

this.showTopNum = data.showTopNum || 0

this.endless = data.endless || 0;

this.needcd = data.needcd || 0;

this.team1 = new PKTeamData({id:1})

this.team2 = new PKTeamData({id:2})

this.team1.enemy = this.team2

this.team2.enemy = this.team1

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

{

var player = new PKPlayerData(data.players[i])

player.teamData = this.getTeamByID(data.players[i].team)

player.teamData.autoDef = Math.max(data.players[i].def || 0,player.teamData.autoDef)

player.teamData.hp += player.hp;

this.playerObj[player.id] = player;

if(player.gameid == UM.gameid)

{

this.myPlayer = player;

player.teamData.atkRota = PKConfig.ROTA\_LEFT

player.teamData.enemy.atkRota = PKConfig.ROTA\_RIGHT

player.teamData.members.unshift(player);

//this.isAuto = player.isauto;

}

else

player.teamData.members.push(player);

}

if(!this.myPlayer) //看别人的录像

{

this.team1.atkRota = PKConfig.ROTA\_LEFT

this.team2.atkRota = PKConfig.ROTA\_RIGHT

this.myPlayer = this.getPlayer(1)

}

this.team1.reInit();

this.team2.reInit();

}

public getForceData(){

var forceObj = {};

for(var s in this.monsterList)

{

var monsterData:PKMonsterData = this.monsterList[s];

var temaID = monsterData.getOwner().teamData.id;

if(!forceObj[temaID])

forceObj[temaID] = 0;

forceObj[temaID] += monsterData.getForce()

}

return forceObj;

}

public random(){

this.randomTimes ++;

var seed = this.randomSeed;

seed = ( seed \* 9301 + 49297 ) % 233280;

var rd = seed / ( 233280.0 );

this.randomSeed = rd \* 100000000;

return rd;

}

public rand(min,max){

return min + Math.floor(this.random()\*(max-min + 1))

}

//数据乱序

public randSort(arr){

var self = this;

arr.sort(rdFun);

function rdFun(){

return self.random()>0.5?-1:1;

}

}

public randomOne(arr:Array<any>,splice = false):any{

var index = Math.floor(arr.length \* this.random())

var data = arr[index];

if(splice)

arr.splice(index,1);

return data;

}

//下一轮开始倒计时

public nextRoundCD(){

var cd = PKTool.cdData[this.round].cd\*1000 - this.actionTime;

return cd

}

//开始游戏

public start(){

this.startTime = TM.nowMS()

this.stopTime = 0;

}

public addDiamondMonster(){

this.diamondData = this.addMonster({

force:0,

mid:99,

owner:'sys',

atkRota:1,

x:PKConfig.floorWidth/2 + PKConfig.appearPos,

y:0,

actionTime:0

});

}

public onPosEmpty(player:PKPlayerData){

}

//要保证只是通知改变显示，不能有逻辑

public addVideo(data){

this.dispatchEventWith('video\_word',false,data)

if(this.quick)

return;

this.dispatchEventWith('video',false,data)

//this.videoList.push(data)

//if(this.topKey.indexOf(data.type) != -1)

// this.topVideoList.push(data);

}

public getTeamByID(teamID){

//if(teamID == 'sys')

// return this.sysTeam;

return this.team1.id == teamID?this.team1:this.team2

}

public getTeamByRota(rota){

return this.team1.atkRota == rota?this.team1:this.team2

}

public getPlayer(id):PKPlayerData{

return this.playerObj[id]

}

public get otherPlayer(){

return this.getPlayer(this.myPlayer.id==1?2:1)

}

//赢

public isWin(){

var team1 = this.myPlayer.teamData

var team2 = this.myPlayer.teamData.enemy

if(this.endless && this.actionTime >= this.endless)

return true;

if(this.needcd && this.actionTime >= this.needcd)

return false;

return team1.hp > 0 && team2.hp <= 0;

}

//平

public isDraw(){

var team1 = this.myPlayer.teamData

var team2 = this.myPlayer.teamData.enemy

if(this.needcd && this.actionTime >= this.needcd)

return false;

return (team1.hp > 0 && team2.hp > 0) || (team1.hp <= 0 && team2.hp <= 0);

}

//赢输平

public getPKResult(){

if(this.isWin())

return 1;

if(this.isDraw())

return 3;

return 2;

}

//找玩家对应的怪

public getMonsterByPlayer(playerid,type=0){

var arr = [];

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

{

var oo = this.monsterList[i];

if(oo.owner == playerid)

{

if(type && oo.getVO().type != type)

continue

arr.push(oo)

}

}

return arr;

}

//找玩家对应的怪的占位

public getMonsterSpaceByPlayer(playerid){

var count = 0;

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

{

var oo = this.monsterList[i];

if(oo.owner == playerid)

{

count += oo.getVO().space;

}

}

return count;

}

//找对应的怪

public getMonsterByID(id){

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

{

var oo = this.monsterList[i];

if(oo.id == id)

{

return oo;

}

}

return null;

}

//找玩家对应的怪

public getMonsterByTeam(team,fun?,value?){

var arr = [];

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

{

var oo = this.monsterList[i];

if(oo.getOwner().teamData == team)

{

if(fun && !fun(oo,value))

continue;

arr.push(oo)

}

}

return arr;

}

//找玩家对应的怪

public getMonsterByNoTeam(team,fun?,value?){

var arr = [];

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

{

var oo = this.monsterList[i];

if(oo.getOwner().teamData != team)

{

if(fun && !fun(oo,value))

continue;

arr.push(oo)

}

}

return arr;

}

//取队伍的最前的怪

public getFirstItem(taamID){

var atkRota = PKData.getInstance().getTeamByID(taamID).atkRota;

var chooseItem

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

{

var item:PKMonsterData = this.monsterList[i];

if(item.atkRota != atkRota)

continue

if(item.owner == 'sys')

continue

if(!chooseItem)

chooseItem = item;

else if(atkRota == PKConfig.ROTA\_LEFT && chooseItem.x<item.x)

chooseItem = item;

else if(atkRota == PKConfig.ROTA\_RIGHT && chooseItem.x>item.x)

chooseItem = item;

}

return chooseItem

}

//取队伍的后排的怪

public getBackItem(taamID){

var atkRota = PKData.getInstance().getTeamByID(taamID).atkRota;

var chooseItem

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

{

var item:PKMonsterData = this.monsterList[i];

if(item.atkRota != atkRota)

continue

if(item.owner == 'sys')

continue

if(!chooseItem)

{

if(!item.getVO().isNearAtk())

chooseItem = item;

}

else if(atkRota == PKConfig.ROTA\_LEFT && chooseItem.x<item.x)

{

if(!item.getVO().isNearAtk())

chooseItem = item;

}

else if(atkRota == PKConfig.ROTA\_RIGHT && chooseItem.x>item.x)

{

if(!item.getVO().isNearAtk())

chooseItem = item;

}

}

return chooseItem

}

public getFirstX(teamID){

var item = this.getFirstItem(teamID);

if(item)

return item.x;

return this.getTeamByID(teamID).atkRota == PKConfig.ROTA\_LEFT ? PKConfig.appearPos:PKConfig.floorWidth + PKConfig.appearPos;

}

//加入怪到场上

public addMonster(data){

var player = this.getPlayer(data.owner)

var toFront = player.teamData.toFront

if(toFront > 0)

{ var frontItem:any = this.getFirstItem(player.teamData.id);

if(frontItem)

{

data.x = frontItem.x;

if(!MonsterVO.getObject(data.mid).isNearAtk())

{

data.x += player.teamData.atkRota == PKConfig.ROTA\_LEFT?-60:60

}

}

}

var monster = new PKMonsterData(data)

monster.id = this.monsterID;

this.monsterID ++;

this.monsterList.push(monster);

this.addVideo({

type:PKConfig.VIDEO\_MONSTER\_ADD,

user:monster

})

MBase.getData(monster.mid).onCreate(monster);

monster.getOwner().teamData.testState(PKConfig.LISTENER\_CREATE,monster);

if(data.fromPos)

monster.getOwner().teamData.testState(PKConfig.LISTENER\_CREATE\_POS,monster);

this.monsterChange = true;

this.history[monster.id] = ({

id:monster.id,

mid:monster.mid,

owner:monster.owner,

actionTime:this.actionTime

})

monster['xxx'] = this.actionTime + '|' + monster.id

this.actionRecord.push('create|' + this.actionTime + '|' + monster.id + '|' + monster.mid+ '|' + monster.owner)

return monster;

}

//重置战场上的怪的数据

public resetMonsterData(){

if(!this.monsterChange)

return;

this.monsterChange = true;

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

{

}

}

public changeMyPlayer(id){

var player = this.getPlayer(id);

if(player == this.myPlayer)

return;

this.myPlayer = player;

this.addVideo({

type:PKConfig.VIDEO\_MYPLAYER\_CHANGE,

user:this.myPlayer

})

}

}

//场上的怪

class PKMonsterData {

public die = false;

public passEnd = false;//冲过了终点

public index//生成的序号

public hp = 0

public atk = 0

public speed = 0

public def = 0

public maxHp = 0

public hpChange = 0 //每秒改变的HP值,是1秒，处理时会除2

public lastHpChange = 0 //上次改变的HP值的时间

public atkAble = true //可以攻击

public level = 0 //英雄的等级

//隐藏属性

public doubleRate = 0

public doubleValue = 0

public doubleAction = false;

public missRate = 0

public isReborn = false

public momian = false

public skillTemp = {}//用于存放技能的临时变量

public nohitTimes = 0//不受伤害的次数

public baseHp = 0

public baseAtk = 0

public baseSpeed = 0

public addSpeed = 0//速度改变百分比

public manaHp = 0//魔盾

private skillTimes=0//技能使用的次数

public x;//当前的位置

public y;//当前的位置

public atkY = 0;//攻击发出的Y值偏移

public atkX = 0;//攻击发出的X值偏移

public mid //对应的怪

public owner//属于哪个玩家

public actionTime//上次行动的时间

public atkRota//进攻方向

public target:PKMonsterData//攻击目标

public skillTargets//技能目标

public id;//唯一ID

public atkHurtCount = 0;//累计输出伤害

public stopTime = 0

public lastSkill = 0

public dieTime = 0

public buff = [];

public currentState = {};//当前的特殊状态

public stateChange = false

public reborning = false//有复活效果起作用了

public stopReborn = false//禁止目标复活

public listenerData//保存用于listener的数据

public callHeroSkill//本次的英雄技能ID

public useingHeroSkill//已触发的英雄技能ID

constructor(obj?){

if(obj)

this.fill(obj);

}

public fill(obj)

{

for (var key in obj) {

this[key] = obj[key];

}

var mvo = MonsterVO.getObject(this.mid);

var add = mvo.getAdd(obj.force,this.getOwner().type);

var maxAdd = obj.maxHpAdd || 1;

this.hp = Math.floor(mvo.hp \* add \* maxAdd);

this.atk = Math.floor(mvo.atk \* add);

this.speed = mvo.speed;

this.def = mvo.def;

this.maxHp = this.hp;

this.baseHp = this.hp;

this.baseAtk = this.atk;

this.baseSpeed = this.speed;

if(obj.hpRate)

{

this.hp = Math.floor(this.hp\*obj.hpRate) || 1

}

MBase.getData(this.mid).initMonster(this);

}

public getForce(){

var mvo = this.getVO();

return Math.round((this.hp/mvo.hp\*(1+this.def/100) + this.atk/mvo.atk\*0.8 + this.speed/mvo.speed\*0.5)\*mvo.cost\*10);

}

//根据属性相克，取攻击比例

public getAtkRate(defender:PKMonsterData){

if(\_get['debug'])

return 1//先不考滤相克

var atkType = this.getVO().type

var defType = defender.getVO().type

return PKTool.getAtkRate(atkType,defType)

}

public getAtkBeforeCD(){

if(!this.addSpeed)

return this.getVO().mv\_atk

if(this.addSpeed < 0)

return Math.floor(this.getVO().mv\_atk\*(1-this.addSpeed/100));

return Math.floor(this.getVO().mv\_atk\*(1-this.addSpeed/(100 + this.addSpeed)));

}

public getAtkCD(){

if(!this.addSpeed)

return this.getVO().atkcd

if(this.addSpeed < 0)

return Math.floor(this.getVO().atkcd\*(1-this.addSpeed/100));

return Math.floor(this.getVO().atkcd\*(1-this.addSpeed/(100 + this.addSpeed)));

}

public beSkillAble(){

return this.momian == false && !this.isInState(PKConfig.STATE\_MOMIAN);

}

//{endTime, add:{属性名称:增加值}， state:{状态名：true}, id:唯一ID, no:这BUFF没生效, value:技能等级数值}

public addBuff(data:PKBuffData){

data.owner = this;

this.buff.push(data);

if(data.id)

this.resetBuffID(data.id);

else

data.enable();

if(data.ing && data.haveState)

this.resetState();

}

//拥有指定BUFF

public haveBuff(id){

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

{

var oo:PKBuffData = this.buff[i];

if(oo.id == id)

{

return true

}

}

return false;

}

//清除状态

public cleanBuff(t,user?,buffType=0){

var needTestStat = false

var needTestId = null

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

{

var oo:PKBuffData = this.buff[i];

var needClean = oo.endTime && t && oo.endTime <= t

if(user && user == oo.user)

needClean = true;

if(oo.removeAble)

{

if(buffType == -1 && oo.isDebuff)

needClean = true;

if(buffType == 1 && !oo.isDebuff)

needClean = true;

}

if(needClean)

{

this.buff.splice(i,1);

i--;

if(oo.ing)

{

if(oo.haveState)

needTestStat = true;

if(oo.id)

{

if(!needTestId)

needTestId = {};

if(!needTestId[oo.id])

needTestId[oo.id] = true;

}

}

oo.remove();

}

}

if(needTestId)

{

for(var s in needTestId)

this.resetBuffID(s);

}

if(needTestStat)

this.resetState();

}

//判断是否在某个状态中

public isInState(stateName){

if(stateName == PKConfig.STATE\_MOMIAN && this.momian)

return true;

return this.currentState[stateName];

}

//重置状态

public resetState(){

var lastState = this.currentState;

this.currentState = {};

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

{

var oo:PKBuffData = this.buff[i];

if(oo.ing && oo.haveState)

{

for(var s in oo.state)

{

this.currentState[s] = true;

if(!lastState[s]) //新增了状态

this.stateChange = true;

}

}

}

if(!this.stateChange)

{

for(var s in lastState)

{

if(!this.currentState[s]) //去除了状态

{

this.stateChange = true;

break;

}

}

}

}

//对ID唯一的技能进行重置

public resetBuffID(id){

var ids = []//所有相同ID的BUFF

var current:PKBuffData;

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

{

var oo:PKBuffData = this.buff[i];

if(oo.id == id)

{

ids.push(oo);

if(oo.ing)

current = oo

}

}

if(!ids.length) //没技能

return

if(ids.length == 1 && current) //只有唯一技能

return

var newOne:PKBuffData;

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

{

var oo:PKBuffData = ids[i];

if(!newOne || Math.abs(oo.value) > Math.abs(newOne.value))

newOne = oo;

}

if(!current) //以前没有

{

newOne.enable();

return

}

if(newOne != current && newOne.value != current.value)

{

newOne.enable();

current.disable();

}

}

public getVO():MonsterVO{

return MonsterVO.getObject(this.mid);

}

public getOwner(){

return PKData.getInstance().getPlayer(this.owner);

}

public canMove(t){

if(this.owner == 'sys')

return false;

if(this.getVO().speed <=0)

return false;

if(!this.canAction())

return false

if(!this.atkAble && PKData.getInstance().currentState == 'def'){

if(this.atkRota == PKConfig.ROTA\_LEFT && this.x > PKConfig.floorWidth/2 + PKConfig.appearPos - 100)

return false

if(this.atkRota == PKConfig.ROTA\_RIGHT && this.x < PKConfig.floorWidth/2 + PKConfig.appearPos + 100)

return false

}

return this.stopTime < t;

}

//可以有行为 如移动，攻击等

public canAction(){

return !this.die && !this.isInState(PKConfig.STATE\_YUN)

}

public canAtk(){

var PD = PKData.getInstance();

return this.atkAble && this.canAction() && this.stopTime < PD.actionTime

}

public canBeAtk(user){

return !this.die &&

user.getOwner().teamData != this.getOwner().teamData && !this.isInState(PKConfig.STATE\_NOBEATK)

}

//可以用技能

public canSkill(t){

if(this.stopTime >= t)

return null;

var vo = this.getVO();

if(!vo.skillcd) //无技能

return null;

if(this.lastSkill && (this.lastSkill + vo.skillcd >= t))

return null;

if(!this.canAction())

return null;

this.skillTargets = MBase.getData(this.mid).getSkillTarget(this);

return this.skillTargets

}

public isHero(){

return this.getVO().isHero();

}

public setSkillUse(actionTime){

if(this.getVO().isHero())

{

if(this.callHeroSkill)

this.useingHeroSkill = this.callHeroSkill;

this.callHeroSkill = 0;

MBase.getData(this.mid).setHeroSkillUse(this,this.useingHeroSkill)

}

if(this.getVO().skillcd < 0)

{

this.skillTimes += 1000;

if(this.getVO().skillcd + this.skillTimes >= 0)

this.lastSkill = Number.MAX\_VALUE;

else

this.lastSkill = actionTime;

}

else

this.lastSkill = actionTime;

}

public move(){

if(!this.addSpeed)

var rate = 1;

else if(this.addSpeed > 0)

var rate = (1+this.addSpeed/100);

else

var rate = (1+this.addSpeed/(100 - this.addSpeed));

this.x += this.atkRota \* Math.round(this.speed\*rate)/10;

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_MOVE,

user:this

})

}

public stand(){

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_STAND,

user:this

})

}

public getAtkTarget(){

if(this.owner == 'sys')

return null;

var PD = PKData.getInstance();

if(!this.canAtk())

return null;

var atkRage = this.getVO().getAtkDis();

if(this.target)

{

if(this.target.canBeAtk(this) && Math.abs(this.target.x - this.x) < atkRage + this.target.getVO().width/2)

{

return this.target;

}

else

this.target = null;

}

//找最近的

var des = 0;

var list = PD.monsterList

var myPlayer = PD.getPlayer(this.owner);

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

{

var target:PKMonsterData = list[i];

var ePlayer = PD.getPlayer(target.owner);

if(myPlayer.teamData.id == ePlayer.teamData.id)//同一队

continue;

var tDes = Math.abs(target.x - this.x);

if(tDes < atkRage + target.getVO().width/2 && target.canBeAtk(this)) {

if (!this.target || des > tDes)

{

this.target = target;

des = tDes;

}

}

}

return this.target

}

//统计累计伤害

public addAtkHurt(v){

if(v > 0)

this.atkHurtCount += v;

}

public atkAction(data){

this.listenerData = data;

if(data.hp)

this.addAtkHurt(data.hp)

this.getOwner().teamData.testState(PKConfig.LISTENER\_ATK,this);

}

public beAtkAction(data){

this.addHp(-data.hp)

MBase.getData(this.mid).beAtkAction(this,data);

this.listenerData = data;

this.getOwner().teamData.testState(PKConfig.LISTENER\_BEATK,this);

if(this.die && data.atker)

{

MBase.getData(data.atker.mid).onKill(data.atker,this);

}

}

//直接死亡

public setDie(){

this.hp = 0;

this.die = true;

}

public addHp(hp){

if(hp<0)

{

if(this.nohitTimes)

{

this.nohitTimes -- ;

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_NOHIT,

user:this,

})

return;

}

else if(this.manaHp)

{

this.manaHp += hp;

if(this.manaHp < 0)

{

hp = this.manaHp

this.manaHp = 0

}

else

{

hp = 0;

}

//魔盾消失

if(this.manaHp == 0)

{

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MANAHP\_CHANGE,

user:this,

})

}

}

if(!hp)

return;

}

this.hp += hp;

if(this.hp <= 0)

this.die = true;

else if(this.hp > this.maxHp)

this.hp = this.maxHp

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_HPCHANGE,

user:this,

})

MBase.getData(this.mid).onHpChange(this);

}

public getHpRate(){

return this.hp / this.maxHp

}

public onDie(){

if(!this.passEnd)

{

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

{

var oo:PKBuffData = this.buff[i];

if(oo.ing && oo.haveState && (oo.state[PKConfig.STATE\_DIE]))

{

var id = parseInt((oo.id + '').split('\_').shift())

MBase.getData(id).onBuff(oo);

}

}

MBase.getData(this.mid).onDie(this);

}

this.getOwner().teamData.testState(PKConfig.LISTENER\_DIE,this);

MBase.getData(this.mid).onRemove(this);

PKData.getInstance().actionRecord.push('die|' + PKData.getInstance().actionTime + '|' + this.id + '|' + this.passEnd)

}

public getSkillValue(index,needForce=false){

var PD = PKData.getInstance();

return CM.getCardVO(this.mid).getSkillValue(index,needForce?PD.getPlayer(this.owner).force:0)

}

}

//玩家数据

class PKPlayerData {

public id;//唯一ID

public gameid;

public nick;

public head;

public hp; //城堡的血量

public type//类型

public force;//怪的基础属性

public teamData:PKTeamData //对应队伍

public autoList

public maxPlayer

constructor(obj?){

if(obj)

this.fill(obj);

if(this.nick)

this.nick = Base64.decode(this.nick);

else

this.nick = '守卫者' + this.id;

}

public fill(obj)

{

obj = ObjectUtil.clone(obj);

//console.log(JSON.stringify(obj));

for (var key in obj) {

this[key] = obj[key];

}

this.autoList = obj['autolist'].split(',');

this.maxPlayer = this.autoList.length;

MonsterVO.getObject(this.autoList[0]).preLoad();

//console.log(this.autoList)

}

public getMonsterForce(mid?){

return this.force

}

public addMonster(){

//var PD = PKData.getInstance();

if(this.autoList.length == 0)

return;

var mid = this.autoList.shift();

if(this.autoList[0])

{

MonsterVO.getObject(this.autoList[0]).preLoad();

}

//var owner = this.id

var atkRota = this.teamData.atkRota;

var x = atkRota == PKConfig.ROTA\_LEFT ? PKConfig.appearPos:PKConfig.floorWidth + PKConfig.appearPos;

PKData.getInstance().addMonster({

force:this.force,

mid:mid,

owner:this.id,

atkRota:atkRota,

fromPos:true,

index:this.maxPlayer - this.autoList.length,

x:x,

y:-25 + Math.random()\*50,

actionTime:PKData.getInstance().actionTime

})

}

}

class PKStateListener { //图腾类，会针对状态改变进行触发

public owner:PKMonsterData; //

public endTime = 0; //到这个点时会自动销毁

public mvID = 0;

public stopDieRemove = false; //如果true,即使主人没了也不要移除

public x = 0; //如果有动画，出现的位置

public type//监听类型

// public id//唯一的ID，ID相同时，value值大的会生效

//public value;

constructor(obj?) {

}

// 起作用时会调用的方法

public actionFun(target?:PKMonsterData){

}

// 图腾移除时会调用的方法

public onRemove(){

}

}

//队伍数据

class PKTeamData {

public id //1 or 2 or sys

public atkRota = 0 //进攻方向 0左路出发，1右路出发

public enemy:PKTeamData

public hp = 0 //城堡的血

public def = 0

public autoDef = 0//10 //防守方增加防御

public toFront = 0;//前线出兵

public killNum = 0;//消灭单位数量

public bornNum = 0;//生产单位数量

public members = [];

public posList = []; //全队上阵的顺序

public stateObj = {}; //监听队伍中的状态，触发

constructor(obj?){

if(obj)

this.fill(obj);

//this.autoDef = PKConfig.autoDef;

}

public fill(obj)

{

for (var key in obj) {

this[key] = obj[key];

}

}

public reInit(){

}

public getTeamDef(){

return Math.floor(this.def / 5) + this.autoDef

}

//监听状态

public addStateLister(listener:PKStateListener){

if(!this.stateObj[listener.type])

this.stateObj[listener.type] = [];

this.stateObj[listener.type].push(listener)

//console.log('add')

}

//

public testState(state,target:PKMonsterData){

if(!this.stateObj[state] || this.stateObj[state].length == 0)

return

for(var i=0;i<this.stateObj[state].length;i++)

{

var listener:PKStateListener = this.stateObj[state][i];

listener.actionFun(target)

}

}

//

public removeStateListener(listener:PKStateListener){

var state = listener.type

if(!state || !this.stateObj[state] || this.stateObj[state].length == 0)

return

ArrayUtil.removeItem(this.stateObj[state],listener);

listener.onRemove()

}

//

public removeStateListerByOwner(owner){

for(var state in this.stateObj)

{

for(var i=0;i<this.stateObj[state].length;i++)

{

var listener:PKStateListener = this.stateObj[state][i];

if(!listener.stopDieRemove && listener.owner == owner)

{

this.stateObj[state].splice(i,1);

i--;

listener.onRemove()

}

}

}

}

public onStateTimer(){

for(var state in this.stateObj)

{

for(var i=0;i<this.stateObj[state].length;i++)

{

var listener:PKStateListener = this.stateObj[state][i];

if(listener.endTime && listener.endTime <= PKData.getInstance().actionTime)

{

this.stateObj[state].splice(i,1);

i--;

listener.onRemove()

}

else if(parseInt(state) == PKConfig.LISTENER\_TIMER)

{

this.stateObj[state][i].actionFun()

}

}

}

}

}

class MBase {

private static baseData = {};

public static getData(id):MBase{

if(!this.baseData[id])

{

var myClass = this.getClass(id)

this.baseData[id] = new myClass();

this.baseData[id].id = id;

}

return this.baseData[id];

}

public static getClass(id){

switch (id){

case 1:return M1;

case 2:return M2;

case 3:return M3;

case 4:return M4;

case 5:return M5;

case 6:return M6;

case 7:return M7;

case 8:return M8;

case 9:return M9;

case 10:return M10;

case 11:return M11;

case 12:return M12;

case 13:return M13;

case 14:return M14;

case 15:return M15;

case 16:return M16;

case 17:return M17;

case 18:return M18;

case 31:return M31;

case 32:return M32;

case 33:return M33;

case 34:return M34;

case 35:return M35;

case 36:return M36;

case 37:return M37;

case 38:return M38;

case 39:return M39;

case 40:return M40;

case 41:return M41;

case 42:return M42;

case 43:return M43;

case 44:return M44;

case 45:return M45;

case 46:return M46;

case 47:return M47;

case 48:return M48;

case 61:return M61;

case 62:return M62;

case 63:return M63;

case 64:return M64;

case 65:return M65;

case 66:return M66;

case 67:return M67;

case 68:return M68;

case 69:return M69;

case 70:return M70;

case 71:return M71;

case 72:return M72;

case 73:return M73;

case 74:return M74;

case 75:return M75;

case 76:return M76;

case 77:return M77;

case 78:return M78;

}

}

public mvID1

public mvID2

public mvID3

public id

public type = 'monster'

constructor() {

}

public isHeroSkillCDOK(user,id){

var lastSkillTime = user.skillTemp['hs' + id] || 0;

var cd = user.getVO().getHeroSkill(id).skillcd;

if(user.mid == 102 && user.level >=5) //减CD技能

{

cd = user.getVO().getHeroSkillValue(5,1) \* 1000;

}

if(PKData.getInstance().actionTime > lastSkillTime + cd)

{

return true;

}

return false;

}

public setHeroSkillUse(user,id,t?){

user.skillTemp['hs' + id] = Math.max(t || PKData.getInstance().actionTime, user.skillTemp['hs' + id] || 0);

}

public onHpChange(user:PKMonsterData){

}

public onCreate(user:PKMonsterData){

}

public onDie(user:PKMonsterData){

}

public onRemove(user:PKMonsterData){

}

public onBuff(buff:PKBuffData){

}

//被攻击时的处理

public beAtkAction(user:PKMonsterData,data){

}

//杀死单位的处理

public onKill(user:PKMonsterData,target:PKMonsterData){

}

//预加载

public preload(){

if(this.mvID1)

AtkMVCtrl.getInstance().preLoadMV(this.mvID1)

if(this.mvID2)

AtkMVCtrl.getInstance().preLoadMV(this.mvID2)

if(this.mvID3)

AtkMVCtrl.getInstance().preLoadMV(this.mvID3)

}

//初始化怪物隐藏属性

public initMonster(user:PKMonsterData){

}

////////////////////////////////////////////////////// skill

//取技能目标

public getSkillTarget(user:PKMonsterData){

return [];

}

//伤害飞行时间

protected getSkillArriveCD(user:PKMonsterData,target:PKMonsterData){

return 0;

}

//技能动画

public skillMV(user,target,actionTime,endTime){

AtkMVCtrl.getInstance().mSkillMV(this.id,user,target,actionTime,endTime)

}

//实现技能

public skill(user:PKMonsterData,target){

}

////////////////////////////////////////////////////// atk

//取技能目标

protected getAtkTargets(user:PKMonsterData){

return [user.target];

}

//伤害飞行时间

protected getAtkArriveCD(user:PKMonsterData,target:PKMonsterData){

return 0;

}

//攻击发出时的附加动画，如箭，魔发效果

public atkMV(user,target,actionTime,endTime){

AtkMVCtrl.getInstance().mAtkMV(this.id,user,target,actionTime,endTime)

}

////////////////////////////////////////////////////// other上面的为要处理的函数

//技能前处理（生成技能事件）

public skillBefore(user:PKMonsterData,actionTime){

var endTime = actionTime + this.getAtkMVCD(user)//这个时间后发出攻击时件(前摇)

var targets = user.skillTargets;

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

{

this.sendSkillBefore(user,targets[i],actionTime,endTime)

}

}

//技能发出处理

public skillAction(user:PKMonsterData,target:PKMonsterData,actionTime){

var endTime = actionTime + this.getSkillArriveCD(user,target);

this.sendSkillAction(user,target,actionTime,endTime) //攻击起作用

}

//攻击前处理（生成PK事件）设攻击发出时间，攻击目标选择

public atkBefore(user:PKMonsterData,actionTime){

var endTime = actionTime + this.getAtkMVCD(user)//这个时间后发出攻击时件(前摇)

var targets = this.getAtkTargets(user);

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

{

this.sendAtkBefore(user,targets[i],actionTime,endTime)

}

}

public getAtkMVCD(user:PKMonsterData){

return user.getAtkBeforeCD();

}

//攻击发出，设攻击生效(起作用)时间

public atkAction(user:PKMonsterData,target:PKMonsterData,actionTime){

var endTime = actionTime + this.getAtkArriveCD(user,target);

if(user.doubleRate && PKData.getInstance().random() < user.doubleRate)

{

user.doubleAction = true;

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_DOUBLE,

user:user,

value:user.atk\*user.doubleValue,

})

}

else

user.doubleAction = false;

this.sendAtkAction(user,target,actionTime,endTime) //攻击起作用

}

//a对B攻击到达时的逻辑（攻击正式生效）

public atk(user:PKMonsterData,target:PKMonsterData){

if(!user.doubleAction && target.missRate && PKData.getInstance().random() < target.missRate) //暴击不可闪

{

if(target.die)

return false;

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_MISS,

user:target

})

return false;

}

var hp = this.getAtkHp(user,target);

target.beAtkAction({hp:hp,atker:user})

user.atkAction({hp:hp})

return true;

}

//取攻击力

protected getAtkerAtk(user:PKMonsterData,target:PKMonsterData){

var atk = user.atk \* user.getAtkRate(target);

if(user.doubleAction)

atk \*= user.doubleValue;

return Math.ceil(atk);

}

//取最终伤害

public getAtkHp(user:PKMonsterData,target:PKMonsterData){

var atk = this.getAtkerAtk(user,target);

var teamDef = target.getOwner().teamData.getTeamDef();

var hp = Math.floor(atk \* Math.max(1-(target.def + teamDef)/100,0));

if(hp < 1)

hp = 1;

return Math.ceil(hp);

}

protected sendAtkBefore(user,target,actionTime,endTime){

PKMonsterAction.getInstance().addAtkList({ //到actionTime后根据条件产生攻击事件

type:'atk\_before',

model:this,

user:user,

target:target,

actionTime:actionTime,

endTime:endTime

})

}

protected sendSkillBefore(user,target,actionTime,endTime){

PKMonsterAction.getInstance().addAtkList({ //到actionTime后根据条件产生攻击事件

type:'skill\_before',

model:this,

user:user,

target:target,

actionTime:actionTime,

endTime:endTime

})

}

protected sendAtkAction(user,target,actionTime,endTime){

PKMonsterAction.getInstance().addAtkList({ //到actionTime后根据条件攻击起作用

type:'atk',

user:user,

target:target,

actionTime:actionTime,

endTime:endTime

})

if(!PKData.getInstance().quick)

this.atkMV(user,target,actionTime,endTime)

}

protected sendSkillAction(user,target,actionTime,endTime){

PKMonsterAction.getInstance().addAtkList({ //到actionTime后根据条件产生攻击事件

type:'skill',

user:user,

target:target,

actionTime:actionTime,

endTime:endTime

})

if(!PKData.getInstance().quick)

this.skillMV(user,target,actionTime,endTime)

}

}

class M1 extends MBase{

constructor() {

super();

}

public mvID1 = 103;

public skill(user:PKMonsterData,target){

var hp = Math.ceil(user.getSkillValue(2,true));

target.beAtkAction({hp:hp,atker:user})

user.atkAction({hp:hp})

}

private testTarget(target,val){

if(!target.beSkillAble())

return false;

var des = Math.abs(val[0].x - target.x);

if(des<=val[1])

{

target.temp = des;

return true

}

}

//对最多3个单位进行一次攻击

public getSkillTarget(user:PKMonsterData){

var PD = PKData.getInstance();

var atkrage = user.getVO().getAtkDis() + 50;

var list = PD.getMonsterByTeam(user.getOwner().teamData.enemy,this.testTarget,[user,atkrage]);

var maxNum = user.getSkillValue(1)

if(list.length>maxNum)

{

//ArrayUtil.sortByField(list,['temp','id'],[0,0])

list.length = maxNum;

}

return list;

}

}

class M2 extends MBase{

constructor() {

super();

}

//初始化怪物隐藏属性

public initMonster(user:PKMonsterData){

user.missRate = user.getSkillValue(1)/100;

}

}

class M3 extends MBase{

constructor() {

super();

}

public preload(){

AtkMVCtrl.getInstance().preLoadPNG('monster/enemy3\_attack.png')

}

public atkAction(user:PKMonsterData,target:PKMonsterData,actionTime){

super.atkAction(user,target,actionTime);

//第二次伤害

var endTime = actionTime + this.getAtkArriveCD(user,target)+50;

this.sendAtkAction(user,target,actionTime,endTime) //攻击起作用

//第3次伤害

var endTime = endTime+50;

this.sendAtkAction(user,target,actionTime,endTime) //攻击起作用

}

//伤害飞行时间

protected getAtkArriveCD(user:PKMonsterData,target:PKMonsterData){

return Math.abs(user.x - target.x) + 200;

}

}

class M4 extends MBase{

constructor() {

super();

}

public atk(user:PKMonsterData,target:PKMonsterData){

var b = super.atk(user,target)

if(b && PKData.getInstance().random() < user.getSkillValue(1)/100)

{

var buff = new PKBuffData()

buff.user = user;

buff.isDebuff = true;

buff.addState(PKConfig.STATE\_YUN);

buff.endTime = PKData.getInstance().actionTime + 1000\*user.getSkillValue(2);

target.addBuff(buff)

}

return b;

}

}

class M5 extends MBase {

constructor() {

super();

}

public initMonster(user:PKMonsterData){

user.doubleRate = user.getSkillValue(1)/100;

user.doubleValue = user.getSkillValue(2)/100;

}

}

class M6 extends MBase {

constructor() {

super();

}

public atk(user:PKMonsterData,target:PKMonsterData){

var b = super.atk(user,target);

if(!b)

return false;

//溅射

var isToRight = user.x<target.x

var PD = PKData.getInstance();

var arr = PD.getMonsterByTeam(user.getOwner().teamData.enemy);

var atkRage = user.getVO().getAtkDis() + user.getSkillValue(2);

var hitRate = user.getSkillValue(1)/100;

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

{

var newTarget = arr[i];

if(newTarget == target)

continue;

if(isToRight)

{

if(user.x > newTarget.x)

continue

}

else if(user.x < newTarget.x)

continue

if(!newTarget.canBeAtk(user))

continue;

var tDes = Math.abs(user.x - newTarget.x);

if(tDes > atkRage + newTarget.getVO().width/2)

continue;

var hp = Math.ceil(this.getAtkHp(user,newTarget)\*hitRate);

newTarget.beAtkAction({hp:hp})

user.addAtkHurt(hp)

}

return true;

}

}

class M7 extends MBase {

constructor() {

super();

}

//初始化怪物隐藏属性

public initMonster(user:PKMonsterData){

user.doubleValue = user.getSkillValue(2)/100;

}

public atkAction(user:PKMonsterData,target:PKMonsterData,actionTime){

if(user.getHpRate() < user.getSkillValue(1)/100)

{

user.doubleRate = 1;

}

super.atkAction(user,target,actionTime);

user.doubleRate = 0

}

}

class M8 extends MBase {

constructor() {

super();

}

public mvID1 = 119;

//初始化怪物隐藏属性

public initMonster(user:PKMonsterData){

user.doubleValue = user.getSkillValue(1)/100;

}

public atkAction(user:PKMonsterData,target:PKMonsterData,actionTime){

if(target.dieTime)

{

user.doubleRate = 1;

}

super.atkAction(user,target,actionTime);

user.doubleRate = 0

}

public skill(user:PKMonsterData,target:PKMonsterData){

target.owner = user.owner

target.atkRota = user.atkRota;

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_CHANGE\_TEAM,

user:target

})

}

//转化

public getSkillTarget(user:PKMonsterData){

var PD = PKData.getInstance();

var arr = PD.getMonsterByTeam(user.getOwner().teamData.enemy);

var atkrage = 300;

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

{

var target = arr[i];

if(!target.dieTime)

continue;

if(target.hp>= user.getSkillValue(2,true))

continue;

if(!target.beSkillAble())

continue;

var des = Math.abs(user.x - target.x);

if(des<=atkrage)

{

return [target]

}

}

return [];

}

}

class M9 extends MBase {

constructor() {

super();

}

public onCreate(user:PKMonsterData){

var PD = PKData.getInstance();

var arr = PD.getMonsterByTeam(user.getOwner().teamData);

var rate = user.getSkillValue(1)/100;

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

{

var target = arr[i];

var buff = new PKBuffData()

buff.id = 9;

buff.user = user;

buff.addValue('atk',Math.max(1,Math.floor(target.baseAtk\*rate)));

target.addBuff(buff)

if(buff.ing)

{

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_ADD\_STATE,

user:target,

keys:['atk+']

})

}

}

var listener = new M9StateListener();

listener.owner = user;

user.getOwner().teamData.addStateLister(listener)

}

public onRemove(user:PKMonsterData){

var PD = PKData.getInstance();

var arr = PD.getMonsterByTeam(user.getOwner().teamData);

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

{

var target:PKMonsterData = arr[i];

target.cleanBuff(0,user);

}

user.getOwner().teamData.removeStateListerByOwner(user)

}

}

class M9StateListener extends PKStateListener {

public type = PKConfig.LISTENER\_CREATE

constructor() {

super();

}

// 起作用时会调用的方法

public actionFun(target?:PKMonsterData){

var rate = this.owner.getSkillValue(1)/100;

var buff = new PKBuffData()

buff.id = 9;

buff.user = this.owner;

buff.addValue('atk',Math.max(1,Math.floor(target.baseAtk\*rate)));

target.addBuff(buff)

if(buff.ing)

{

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_ADD\_STATE,

user:target,

keys:['atk+']

})

}

}

}

class M10 extends MBase {

constructor() {

super();

}

public mvID1 = 128;

public initMonster(user:PKMonsterData){

user.atkAble = false;

}

public skill(user:PKMonsterData,target:PKMonsterData){

target.addHp(user.getSkillValue(1,true))

}

//治疗1个单位

public getSkillTarget(user:PKMonsterData){

var PD = PKData.getInstance();

var arr = PD.getMonsterByTeam(user.getOwner().teamData);

var atkrage = user.getVO().getAtkDis() + 100;

var selectTarget

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

{

var target = arr[i];

if(target.hp >= target.maxHp)

continue;

var des = Math.abs(user.x - target.x);

if(des<=atkrage)

{

target.temp = target.getHpRate();

if(!selectTarget || selectTarget.temp > target.temp)

selectTarget = target

}

}

if(selectTarget)

{

return [selectTarget]

}

return [];

}

}

class M11 extends MBase {

constructor() {

super();

}

public onCreate(user:PKMonsterData){

user.momian = true

}

}

class M12 extends MBase {

constructor() {

super();

}

public atk(user:PKMonsterData,target:PKMonsterData){

var b = super.atk(user,target)

if(b && target.beSkillAble())

{

var skillValue = user.getSkillValue(1);

var buff = new PKBuffData()

buff.id = 12;

buff.isDebuff = true;

buff.value = skillValue;

buff.addValue('atk',-Math.floor(target.baseAtk \* skillValue/100));

buff.addValue('def',-skillValue);

buff.user = user;

buff.endTime = PKData.getInstance().actionTime + 1000\*user.getSkillValue(2);

target.addBuff(buff)

if(buff.ing)

{

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_ADD\_STATE,

user:target,

keys:['atk-','def-']

})

}

}

return b;

}

}

class M13 extends MBase {

constructor() {

super();

}

public initMonster(user:PKMonsterData){

user.atkAble = false;

}

public onCreate(user:PKMonsterData){

var listener = new M13StateListener();

listener.owner = user;

user.getOwner().teamData.addStateLister(listener)

}

public onRemove(user:PKMonsterData){

user.getOwner().teamData.removeStateListerByOwner(user)

}

}

class M13StateListener extends PKStateListener {

public type = PKConfig.LISTENER\_TIMER

public actionTime

//public x

constructor() {

super();

this.actionTime = PKData.getInstance().actionTime;

}

// 起作用时会调用的方法

public actionFun(target?:PKMonsterData){

if(PKData.getInstance().actionTime - this.actionTime < 500)

return;

this.actionTime = PKData.getInstance().actionTime;

var user:PKMonsterData = <PKMonsterData>this.owner;

var PD = PKData.getInstance();

var arr = PD.getMonsterByTeam(user.getOwner().teamData.enemy);

var atkrage = user.getSkillValue(1);

var list = [];

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

{

var targetEnemy = arr[i];

if(!targetEnemy.beSkillAble())

continue;

var des = Math.abs(user.x - targetEnemy.x);

if(des<=atkrage)

{

targetEnemy.addHp(-Math.ceil(user.getSkillValue(2,true)\*user.getAtkRate(targetEnemy)\*0.5))

}

}

return list;

}

}

class M14 extends MBase {

constructor() {

super();

}

//被攻击时的处理

public beAtkAction(user,data){

if(data.atker && data.atker.getVO().isNearAtk())

{

var hp = Math.ceil(data.hp\*user.getSkillValue(1)/100)

data.atker.addHp(-hp);

user.addHp(hp)

}

}

}

class M15 extends MBase {

constructor() {

super();

}

//a对B攻击到达时的逻辑（攻击正式生效）

public atk(user:PKMonsterData,target:PKMonsterData){

var b = super.atk(user,target)

if(b)

{

user.atk += Math.ceil(user.baseAtk\*user.getSkillValue(1)/100)

}

return b;

}

}

class M16 extends MBase {

constructor() {

super();

}

//a对B攻击到达时的逻辑（攻击正式生效）

public atk(user:PKMonsterData,target:PKMonsterData){

var b = super.atk(user,target)

if(b)

{

this.addBuff(user,target)

}

return b;

}

//被攻击时的处理

public beAtkAction(user,data){

var target = data.atker

if(target && target.getVO().isNearAtk())

{

this.addBuff(user,target)

}

}

private addBuff(user,target){

if(target.beSkillAble())

{

var skillValue = user.getSkillValue(1,true)

var buff = new PKBuffData()

buff.id = 16;

buff.isDebuff = true;

buff.value = skillValue

buff.addValue('hpChange',-skillValue);

buff.user = user;

buff.endTime = PKData.getInstance().actionTime + 1000\*user.getSkillValue(2);

target.addBuff(buff)

if(buff.ing)

{

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_ADD\_STATE,

user:target,

keys:['hp-']

})

}

}

}

}

class M17 extends MBase {

constructor() {

super();

}

//被攻击时的处理

public beAtkAction(user,data){

//{hp:hp,atker:user}

if(data.atker)

data.atker.addHp(-Math.ceil(user.getSkillValue(1,true)\*user.getAtkRate(data.atker)));

}

}

class M18 extends MBase {

constructor() {

super();

}

//初始化怪物隐藏属性

public initMonster(user:PKMonsterData){

user.hpChange += user.getSkillValue(1,true);

}

}

class M31 extends MBase {

constructor() {

super();

}

public initMonster(user:PKMonsterData){

user.atkY = -30

}

//伤害飞行时间

protected getAtkArriveCD(user:PKMonsterData,target:PKMonsterData){

return Math.abs(user.x - target.x) + 100;

}

public atk(user:PKMonsterData,target:PKMonsterData){

var b = super.atk(user,target)

if(b && PKData.getInstance().random() < user.getSkillValue(1)/100)

{

var buff = new PKBuffData()

buff.user = user;

buff.isDebuff = true;

buff.addState(PKConfig.STATE\_YUN);

buff.endTime = PKData.getInstance().actionTime + 1000\*user.getSkillValue(2);

target.addBuff(buff)

}

return b;

}

}

class M32 extends MBase {

constructor() {

super();

}

public initMonster(user:PKMonsterData){

user.atkY = -30

}

//伤害飞行时间

protected getAtkArriveCD(user:PKMonsterData,target:PKMonsterData){

return Math.abs(user.x - target.x) + 100;

}

public onCreate(user:PKMonsterData){

var listener = new M32StateListener();

listener.owner = user;

user.getOwner().teamData.addStateLister(listener)

}

public onRemove(user:PKMonsterData){

user.getOwner().teamData.removeStateListerByOwner(user)

}

}

class M32StateListener extends PKStateListener {

public type = PKConfig.LISTENER\_TIMER

public actionTime

constructor() {

super();

this.actionTime = PKData.getInstance().actionTime;

}

// 起作用时会调用的方法

public actionFun(target?:PKMonsterData){

if(PKData.getInstance().actionTime - this.actionTime < 1000)

return;

this.actionTime = PKData.getInstance().actionTime;

var user:PKMonsterData = <PKMonsterData>this.owner;

var PD = PKData.getInstance();

var arr = PD.getMonsterByTeam(user.getOwner().teamData);

var atkrage = user.getSkillValue(1);

var list = [];

var hp = user.getSkillValue(2,true);

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

{

var targetEnemy = arr[i];

var des = Math.abs(user.x - targetEnemy.x);

if(des<=atkrage)

{

targetEnemy.addHp(hp)

}

}

return list;

}

}

class M33 extends MBase {

constructor() {

super();

}

public initMonster(user:PKMonsterData){

user.atkY = -30

}

//伤害飞行时间

protected getAtkArriveCD(user:PKMonsterData,target:PKMonsterData){

return Math.abs(user.x - target.x) + 100;

}

public atk(user:PKMonsterData,target:PKMonsterData){

var hp = Math.floor((1-target.getHpRate())\*target.maxHp\*0.15)

var b = super.atk(user,target)

if(b && target.mid != 99 && hp > 0)

{

hp = Math.min(hp,user.getSkillValue(1,true));

target.beAtkAction({hp:hp})

user.addAtkHurt(hp)

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_DOUBLE,

user:user,

value:hp,

})

}

return b;

}

}

class M34 extends MBase {

constructor() {

super();

}

public initMonster(user:PKMonsterData){

user.atkY = -30

}

//伤害飞行时间

protected getAtkArriveCD(user:PKMonsterData,target:PKMonsterData){

return Math.abs(user.x - target.x) + 100;

}

public atk(user:PKMonsterData,target:PKMonsterData){

var b = super.atk(user,target)

if(b && target.beSkillAble())

{

var skillValue = user.getSkillValue(1);

var buff = new PKBuffData()

buff.id = 34;

buff.isDebuff = true;

buff.value = skillValue;

buff.addValue('addSpeed',-skillValue);

buff.user = user;

buff.endTime = PKData.getInstance().actionTime + 1000\*user.getSkillValue(2);

target.addBuff(buff)

if(buff.ing)

{

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_ADD\_STATE,

user:target,

keys:['speed-']

})

}

}

return b;

}

}

class M35 extends MBase {

constructor() {

super();

}

public mvID1 = 103;

public initMonster(user:PKMonsterData){

user.atkY = -30

}

//伤害飞行时间

protected getAtkArriveCD(user:PKMonsterData,target:PKMonsterData){

return Math.abs(user.x - target.x) + 100;

}

public skill(user:PKMonsterData,target:PKMonsterData){

var PD = PKData.getInstance();

var mid = 35;

var owner = PD.getPlayer(user.owner);

var atkRota = owner.teamData.atkRota;

var num =1

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

{

var mData = {

force:owner.getMonsterForce(mid),

mid:mid,

owner:user.owner,

atkRota:atkRota,

x:user.x,

y:-25 + Math.random()\*50,

lastSkill:Number.MAX\_VALUE,

dieTime:PD.actionTime + user.getSkillValue(1)\*1000,

actionTime:PD.actionTime

}

PD.addMonster(mData);

}

}

public getSkillTarget(user:PKMonsterData){

user.getAtkTarget();

if(user.target)

return [null];

return [];

}

}

class M36 extends MBase {

constructor() {

super();

}

public mvID1 = 119;

public onDie(user:PKMonsterData){

if(user.skillTemp[36])

return;

if(user.reborning)

return;

user.reborning = true;

var PD = PKData.getInstance();

PKMonsterAction.getInstance().addAtkList({ //到actionTime后根据条件产生攻击事件

type:'skill\_before',

model:this,

user:user,

target:user,

stopTestDie:true,

actionTime:PD.actionTime,

endTime:PD.actionTime + 1000\*user.getSkillValue(1)

})

}

protected sendSkillAction(user,target,actionTime,endTime){

PKMonsterAction.getInstance().addAtkList({ //到actionTime后根据条件产生攻击事件

type:'skill',

user:user,

target:target,

stopTestDie:true,

actionTime:actionTime,

endTime:endTime

})

}

public skill(user:PKMonsterData,targets){

var PD = PKData.getInstance();

var mid = 36;

var owner = PD.getPlayer(user.owner);

var atkRota = owner.teamData.atkRota;

var mData = {

force:owner.getMonsterForce(mid),

mid:mid,

owner:user.owner,

atkRota:atkRota,

x:user.x,

y:user.y,

index:user.index,

isReborn:true,

lastSkill:Number.MAX\_VALUE,

actionTime:PD.actionTime

}

var monster = PD.addMonster(mData);

monster.skillTemp[36] = true;

var mc = AtkMVCtrl.getInstance().playAniOn(monster.id,this.mvID1)

if(mc)

{

mc.y -= 30

}

}

}

class M37 extends MBase {

constructor() {

super();

}

public atk(user:PKMonsterData,target:PKMonsterData){

var b = super.atk(user,target);

if(!b)

return false;

//溅射

var isToRight = user.x<target.x

var PD = PKData.getInstance();

var arr = PD.getMonsterByTeam(user.getOwner().teamData.enemy);

var atkRage = user.getVO().getAtkDis() + user.getSkillValue(2);

var hitRate = user.getSkillValue(1)/100;

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

{

var newTarget = arr[i];

if(newTarget == target)

continue;

if(isToRight)

{

if(user.x > newTarget.x)

continue

}

else if(user.x < newTarget.x)

continue

if(!newTarget.canBeAtk(user))

continue;

var tDes = Math.abs(user.x - newTarget.x);

if(tDes > atkRage + newTarget.getVO().width/2)

continue;

var hp = Math.ceil(this.getAtkHp(user,newTarget)\*hitRate);

newTarget.beAtkAction({hp:hp})

user.addAtkHurt(hp)

}

return true;

}

}

class M38 extends MBase {

constructor() {

super();

}

public initMonster(user:PKMonsterData){

user.atkY = 10

user.atkX = 30

}

//伤害飞行时间

protected getAtkArriveCD(user:PKMonsterData,target:PKMonsterData){

return Math.abs(user.x - target.x)\*2 + 100;

}

protected getSkillArriveCD(user:PKMonsterData,target:PKMonsterData){

return Math.abs(user.x - target.x)\*2 + 100;

}

public skill(user:PKMonsterData,target){

var hp = Math.ceil(this.getAtkHp(user,target));

target.beAtkAction({hp:hp,atker:user})

user.atkAction({hp:hp})

var skillValue = user.getSkillValue(1);

var buff = new PKBuffData()

buff.isDebuff = true;

buff.value = skillValue;

buff.addValue('def',-skillValue);

buff.addValue('addSpeed',-skillValue);

buff.addValue('atk',-Math.floor(skillValue/100\*target.baseAtk));

buff.user = user;

buff.endTime = PKData.getInstance().actionTime + 1000\*user.getSkillValue(2);

target.addBuff(buff)

if(buff.ing)

{

PKData.getInstance().addVideo({

type:PKConfig.VIDEO\_MONSTER\_ADD\_STATE,

user:target,

keys:['def-','speed-','atk-']

})

}

}

//对最多3个单位进行一次攻击

public getSkillTarget(user:PKMonsterData){

user.getAtkTarget();

var target = user.target

if(target && target.beSkillAble())

{

return [target];

}

return [];

}

}