# 拳皇帧事件添加说明



## 说明

分为英雄主骨骼帧事件添加和英雄子骨骼帧事件添加。主骨骼和子骨骼为独立的两个骨骼动画，因此需要分开添加。例如英雄roleK为主骨骼，释放技能S+D+J时会生成子骨骼fire1。

## 层次说明

role类中的mc为二层骨骼。可执行动作为mc.gotoandStop("punch"/"land"/"jump"/...)。该骨骼无帧事件

role类中的bodyMC为三层骨骼。为mc进入某动作后的骨骼。该骨骼为正常帧事件

bodyMC的children拥有单帧事件

## 帧事件检查顺序

```
/** 帧事件检测
     * 检测顺序:@1:增加每个动作的帧事件
     * @2:检测并执行三层骨骼的帧事件
     * @3:检测并执行单帧事件
     * 注:三层骨骼bodyMC中的帧事件会初始化单帧类属性值
    */
    public checkEvent() {
        EventUtil.checkFrameEvent(this.bodyMC, "roleK", this);
        this.bodyMC.checkFrameEvent();
        EventUtil.checkSingleFrameEvent(this.bodyMC, "roleK", this);
    }
```

## 帧事件处理类EventUtil

静态成员Roles和skill_effects起到筛选作用。

```
class EventUtil {
public static Roles: Array<string> = ["roleK"];
public static skill_effects: Array<string> = ["roleK_skill"];
}
```

### EventUtil.checkFrameEvent()函数

为骨骼动画bodyMC增加每个动作的帧事件

EventroleK的event形如[["s5146", K_dash.addSprite5146],["s5296", K_land.addSprite5296],...]

遍历event，取bodyMC当前动作标签frameLabel("s5146","s5296"...)进行对比，若相同则调用对应的函数实现帧事件添加

```
public static checkFrameEvent(mc: zmovie.ZMovieClip, roleName: string, own_role:
Role) {
    if (EventUtil.Roles.indexOf(roleName) == -1 &&
EventUtil.skill_effects.indexOf(roleName) == -1) {
        return;
    }
    let EventClass = egret.getDefinitionByName("Event" + roleName);
    ////////////////////////////////////////////////////////////////////////////
    //此处生成的class可为EventroleK/EventroleRyu...
    //主骨骼事件类命名规范为"Event" + roleName
    ////////////////////////////////////////////////////////////////////////////
    if (mc.frameLabel == mc._mcName) {
        return;
    }
    // Role的事件处理调用(第二层级的骨骼帧事件,如walk,punch等动作骨骼)
    for (let i = 0; i < EventClass.event.length; i++) {
        if (mc._mcName == EventClass.event[i][0]) {
            EventClass.event[i][1].call(mc, mc, own_role);
        }
    }
}
```

**bodyMC.checkFrameEvent();**

执行bodyMC所拥有的帧事件

**EventUtil.checkSingleFrameEvent()函数**

执行bodyMC的children所拥有的帧事件

```
/**通过遍历MC的children,从而处理子骨骼下只有一帧的事件 */
public static checkSingleFrameEvent(MC: zmovie.ZMovieClip, roleName: string,
own_role: Role) {
    let EventClass = egret.getDefinitionByName("Event" + roleName);
    for (let i = 0; i < MC.$children.length; i++) {
        if (MC.$children[i] instanceof zmovie.ZMovieClip) {
            let mcName: string = (<zmovie.ZMovieClip>MC.$children[i])._mcName;
            if (EventClass.singleFrameEvent[mcName]) {
                EventClass.singleFrameEvent[mcName].call(MC,
<zmovie.ZMovieClip>MC.$children[i], own_role);
            }
        }
    }
}
```

**英雄事件类EventroleK**

命名规则:Event+_roleName

```
/**roleK的主骨骼帧事件处理类
 * 用于实现roleK的帧事件检查
 * event用于增加第二层级帧事件
 * singleFrameEvent用于增加第三层级且单帧事件
 */
class EventroleK {

    //第三层级,单帧事件
    public static singleFrameEvent = {
        "a985": K_a985.addEventa985,
        "a917": K_a917.addEventa917,
        ...
    };
    //第二层级
    public static event = [
        ["s5146", K_dash.addSprite5146],
        ["s5296", K_land.addSprite5296],
        ...
    ];
}
```

# 主骨骼增加帧事件举例(为roleK增加punch动作的帧事件)

**步骤1 新建K_punch类并加入帧事件**

```
class K_punch {

  public static own_role: Role;
  public static mc: zmovie.ZMovieClip;
  public static addSprite5367(mc: zmovie.ZMovieClip, own_role: Role) {
```

```
    this.own_role = own_role;
    this.mc = mc;
    mc.addFrameScript(mc, [1, K_punch.frame1, 4, K_punch.frame4, 5,
K_punch.frame5, 22, K_punch.frame22]);
    return;
  }

  public static frame1() {
    K_a742.getInstance(this.own_role).flag = "punch2";
  }


  public static frame4() {
    //_level0.p_snd.start();
  }

  public static frame5() {
    let hitcheck = K_HitCheck.getInstance(this.own_role);
    hitcheck.hurtBack = "hurt1";
    hitcheck.hurtAway = "hurt6";
    hitcheck.flag = 1;
    hitcheck.end_status = "";
    hitcheck.Vx = 20;
    hitcheck.Vy = 0;
    hitcheck.a = 3;
    hitcheck.g = 0;
    hitcheck.freezeTime2 = 0;
    hitcheck.freezeTime = 3;
    hitcheck.slowTime = 0;
    hitcheck.shake = 0;
    hitcheck.d_rate = 1;
    hitcheck.isObj = false;
    hitcheck.hitType = "hitFire1";
    hitcheck.hitEff = "";
    hitcheck.hitShinning = false;
    hitcheck.reCheck = true;
    hitcheck.checkTimes = 1;
    hitcheck.hitPos = "";
    return;
  }

  public static frame22() {
    // EventUtil.printMC(this.mc, "5367_22");
    this.mc.stop();
    this.own_role.toStatus_switch("stand");
  }

}
```

**步骤2 EventroleK.event.push(["s5367", K_punch.addSprite5367])**

## 主骨骼单帧事件举例(为roleK的a985子骨骼增加单帧事件)

**步骤1 新建K_a985事件类并加入帧事件**

```
class K_a985 {

    public s_type: string;
    public f_time: number;
    public bgType: string;
    public roleName: string;
    public faceName: string;
    public static Firstowner: Role;
    public static instance_First: K_a985;
    public static instance_Second: K_a985;

    constructor() {
        return;
    }

    /**由于会出现K VS K的情况，因此子骨骼使用双例模式，使用own_role作为参数与bodyMC进行绑
定。*/
    public static getInstance(owner: Role): K_a985 {
        if (K_a985.Firstowner == null) {
            K_a985.Firstowner = owner;
            K_a985.instance_First = new K_a985();
            return K_a985.instance_First;
        } else if (K_a985.Firstowner == owner) {
            return K_a985.instance_First;
        } else {
            if (K_a985.instance_Second == null) {
                K_a985.instance_Second = new K_a985();
            }
            return K_a985.instance_Second;
        }
    }

    public static addEventa985(MC: zmovie.ZMovieClip, own_role: Role) {
        MC.visible = false;
        let a917 = K_a985.getInstance(own_role);
        MC.stop();
        MC.visible = false;
        let dx = own_role.x + MC.x * own_role.dir;
        let dy = own_role.y + MC.y;
        //
_level0.effect_mc.superStart(s_type,f_time,dx,dy,bgType,roleName,dir,this,role);
    }
}
```

**步骤2 EventroleK.singleFrameEvent加入{"a985": K_a985.addEventa985,}**

注：二层骨骼帧事件中会出现on(constructor)初始化三层骨骼属性，此时使用getInstance()取得与其绑定的三层骨骼

# 子骨骼增加帧事件举例(增加fire1帧事件)

添加方法类似于主骨骼

### 步骤1 新建子骨骼帧事件处理类EventroleK_skill，等同于EventroleK

```
/**roleK的子骨骼帧事件处理类
 * 用于实现roleK子骨骼,即addskill()...中生成的骨骼帧事件检查
 * 命名:Event+_roleName + _skill
 * event用于增加第二层级帧事件
 * singleFrameEvent用于增加第三层级且单帧事件
 */

class EventroleK_skill {
    //第二层级
    public static event = [
        ["fire1", fire1.addSprite4857],
        ["fire2", fire2.addSprite4772],
        ...
    ];

    //第三层级,单帧事件
    public static singleFrameEvent = {
        "HitCheck_obj": Hero_HitCheck_obj.addEventHitCheck_obj,
        ...
    };
}
```

### 步骤2 新建fire1类并加入相应帧事件

```
class fire1 {
    public static own_role: Role;
    public static mc: zmovie.ZMovieClip;
    public static addSprite4857(mc: zmovie.ZMovieClip, own_role: Role) {
        fire1.own_role = own_role;
        fire1.mc = mc;
        mc.addFrameScript(mc, [1, fire1.frame1, 3, fire1.frame3, 22,
fire1.frame22]);
        return;
    }

    public static frame1() {
        // _level0.fire2_snd.start();
    }

    public static frame3() {
        let k_hitcheck_obj = Hero_HitCheck_obj.getInstance(fire1.own_role);
        k_hitcheck_obj.obj_Vx = 0;
        k_hitcheck_obj.obj_Vy = 0;
        k_hitcheck_obj.obj_level = 0;
        k_hitcheck_obj.hurtBack = "hurt2";
        k_hitcheck_obj.hurtAway = "hurt3";
        k_hitcheck_obj.flag = 1;
        k_hitcheck_obj.end_status = "";
        k_hitcheck_obj.Vx = 20;
        k_hitcheck_obj.Vy = 0;
```

```
        k_hitcheck_obj.a = 3;
        k_hitcheck_obj.g = 0;
        k_hitcheck_obj.freezeTime2 = 2;
        k_hitcheck_obj.freezeTime = 6;
        k_hitcheck_obj.slowTime = 0;
        k_hitcheck_obj.shake = 0;
        k_hitcheck_obj.d_rate = 1;
        k_hitcheck_obj.isObj = true;
        k_hitcheck_obj.hitType = "hitFire1";
        k_hitcheck_obj.hitEff = "fire_m2";
        k_hitcheck_obj.hitShinning = false;
        k_hitcheck_obj.reCheck = true;
        k_hitcheck_obj.checkTimes = 1;
        k_hitcheck_obj.hitPos = "";
        k_hitcheck_obj.nohit_status = [];
        k_hitcheck_obj.nohit_status[0] = "hurt3";
        k_hitcheck_obj.nohit_status[1] = "hurt4";
    }

    public static frame22() {
        fire1.own_role.removeEffect('fire1');
        fire1.mc.parent.removeChild(fire1.mc);
    }

}
```

**步骤3 EventroleK_skill.event.push([["fire1", fire1.addSprite4857]])**

#####

# 子骨骼单帧事件举例(增加Hero_HitCheck_obj帧事件)

**步骤1 新建Hero_HitCheck_obj类并加入帧事件**

```
class Hero_HitCheck_obj {

    public obj_Vx: number;
    public obj_Vy: number;
    public obj_level: number;
    public hurtBack: string;
    public hurtAway: string;
    public flag: any;
    public end_status: string;
    public Vx: number;
    public Vy: number;
    public a: number;
    public g: number;
    public freezeTime2: number;
    public freezeTime: number;
    public slowTime: number;
    public shake: number;
    public d_rate: number;
    public isObj: boolean;
    public hitType: string;
    public hitEff: string;
    public hitShinning;
    public reCheck: boolean;
```

```
    public checkTimes: number;
    public hitPos: string;
    public nohit_status: Array<string> = [];
    public nohit_skills: Array<string> = [];


    public static Firstowner: Role;
    public static instance_First: Hero_HitCheck_obj;
    public static instance_Second: Hero_HitCheck_obj;

    constructor() {
        return;
    }

    public static getInstance(owner: Role): Hero_HitCheck_obj {
        if (Hero_HitCheck_obj.Firstowner == null) {
            Hero_HitCheck_obj.Firstowner = owner;
            Hero_HitCheck_obj.instance_First = new Hero_HitCheck_obj();
            return Hero_HitCheck_obj.instance_First;
        } else if (Hero_HitCheck_obj.Firstowner == owner) {
            return Hero_HitCheck_obj.instance_First;
        } else {
            if (Hero_HitCheck_obj.instance_Second == null) {
                Hero_HitCheck_obj.instance_Second = new Hero_HitCheck_obj();
            }
            return Hero_HitCheck_obj.instance_Second;
        }
    }

    public static addEventHitCheck_obj(MC: zmovie.ZMovieClip, own_role: Role) {
        let s1 = Hero_HitCheck_obj.getInstance(own_role);
        MC.stop();
        MC.visible = false;
        let hitValue = new HitValue();
        hitValue.area = MC;
        hitValue.hurtBack = s1.hurtBack;
        hitValue.hurtAway = s1.hurtAway;
        hitValue.flag = s1.flag;
        hitValue.end_status = s1.end_status;
        hitValue.Vx = s1.Vx;
        hitValue.Vy = s1.Vy;
        hitValue.a = s1.a;
        hitValue.g = s1.g;
        hitValue.freezeTime2 = s1.freezeTime2;
        hitValue.freezeTime = s1.freezeTime;
        hitValue.slowTime = s1.slowTime;
        hitValue.shake = s1.shake;
        hitValue.d_rate = s1.d_rate;
        hitValue.isObj = s1.isObj;
        hitValue.hitType = s1.hitType;
        hitValue.hitEff = s1.hitEff;
        hitValue.hitShinning = s1.hitShinning;
        hitValue.reCheck = s1.reCheck;
        hitValue.checkTimes = s1.checkTimes;
        hitValue.hitPos = s1.hitPos;
        hitValue.nohit_status = s1.nohit_status;
```

```
        // hitV = {area: s1, Vx: Vx, Vy: Vy, a: a, g: g, hitShinning:
hitShinning, reCheck: reCheck, checkTimes: checkTimes, hitPos: hitPos, hurtBack:
hurtBack, hurtAway: hurtAway, freezeTime: freezeTime, freezeTime2: freezeTime2,
slowTime: slowTime, hitType: hitType, hitEff: hitEff, d_rate: d_rate, flag: flag,
shake: shake, end_status: end_status, isObj: isObj, nohit_status: nohit_status,
nohit_skills: nohit_skills};
        HitCheck_obj.getInstance().start_hitCheck(hitValue, own_role);
    }
}
```

**步骤2 EventroleK_skill.singleFrameEvent加入{"HitCheck_obj": Hero_HitCheck_obj.addEventHitCheck_obj,}**