一.检查阶段

1.BuffCommands检测

参数解析：

若BuffCommands配置不为空，将配置字符串"a;b;c;d",解析出四个字段,分别是

```
public enum ECastSkillByBuff
{
    Invalid=0,
    Cast_Skill_ByBuff =1，//当角色存在特定buff时，转为施放TargetSkillId技能
    Not_Cast_Skill_BuffExist = 2 ,//当角色存在特定buff时，技能才可施放
    Not_Cast_Skill_BuffNotExist =3 ；//当角色不存在特定buff时，技能才可施放
}
public enum ECastSkillByBuff ；//Buff命令类型
public int BuffId;  //Buff的ID
public byte BuffLv;  //Buff的等级（若为0，则代表对buff等级不做限制）
public int TargetSkillId; //目标技能ID（只有ECastSkillByBuff==1时才使用）
```

检测流程：

若类型为 Cast_Skill_ByBuffs， 则当角色存在特定buff ，转为施放TargetSkillId技能，原技能停止施放，否则继续施放原技能；

若类型为Not_Cast_Skill_BuffExist，则当角色存在特定buff时，技能才可施放，否则停止技能施放

若类型为Not_Cast_Skill_BuffNotExist，当角色不存在特定buff时，技能才可施放，否则停止技能施放

2.判断技能冷却

若技能处于冷却中，则技能不可施放；

3.PowerConsume检测

参数解析：

PowerConsume最多可配置三个参数，将配置字符串"a;b;c",解析出三个字段,分别是：

int powerConsume； //技能最低耗蓝量 （低于此值，技能不可施放）

int powerConversionRate: //额外技能转换为伤害的比率（如5表示，每5点蓝量可转换为百分之一的额外技能伤害）

int modifyPara ；//附加额外技能伤害修正参数

若配置参数只1个，则技能耗蓝为固定耗蓝；

若配置参数为3个，则技能耗蓝为耗空当前所有蓝量，并按耗蓝的多少对技能造成的伤害进行百分比增强；此时技能伤害增强率:

S=（CurPower -PowerConsume）/ powerConversionRate * (modifyPara /100)

技能最终伤害 = （1+ S ） * 技能基础伤害

PS：技能基础伤害是耗蓝为最低powerConsume的情况下的技能的原本配置伤害

检测流程：

在此阶段，只取powerConsume参数，与当前蓝量进行比较，如蓝量不足，则停止技能施放；

4.判断角色状态是否可以是施放技能

判断流程：

a.若当前正在施放的技能不为空，需要将当前技能中断，中断原因根据下述判断进行初始化；若当前技能为空，则跳过该过程：

b.若角色当前处于受伤动作硬直状态 ，或者处于击倒起身动作状态时，

获取技能配置的BreakBeHitSpasticity参数，

如配置为非0，则当前待施放技能可破受击硬直，中断原因初始化为"破硬直"；

如配置为0，则当前待施放技能不可破受击硬直，不做任何操作；

c. 获取技能配置BreakAttackSpasticity参数，

如配置为0，则待施放技能不可破当前技能硬直，不做任何操作；

如配置为非0，则比较待释放技能和当前技能各自的配置参数BreakSpasticPriority

若待释放技能破硬直权重大于当前技能破硬直权重，则中断原因初始化为"破硬直"

d.读取当前技能的配置字段 SharedCoolDownID，若不为空，则代表当前技能为连击技能，若待施放技能存在于SharedCoolDownID中， 则当前技能和待释放技能属于同一连击系列技能，共享CD，则将中断原因初始为“连击中断”；

e. 根据初始化的中断原因，来中断当前技能；

5.施放技能初始化

1.初始化待释放技能的攻击标识编号（用以区别不同攻击的值，避免同一敌人被同一技能的同一碰撞盒攻击两次）；

2.将技能阶段初始化为：起手段

3.激活碰撞盒；（此时只是激活，并未开始确认目标）

4.应用技能配置参数CanInSky是否为1，来决定是否禁用重力；

5.初始化飞弹信息

6.移除单位身上所有停止条件为“攻击”的buff和光环；

7.检测是否触发QTE；

8.应用单位施放技能对其他单位所带来的仇恨变化；

9.应用技能配置的状态SkillStatus(0：无，1：无敌，2：超级霸体），使单位在硬直结束前一直保持SkillStatus配置的状态；

10.显示技能的击打范围；

11.如果SkillTemplate配置字段IsKillLastFx不为0，则在此时会结束上个技能的特效；

6.技能逻辑更新Update

1.应用技能键伤配置

获取技能的键值增伤参数，并应用到角色属性中；

（键值增伤参数不是在技能表里配置的，而是通过其他途径初始化的）

2.检测技能所有碰撞盒

若技能不处于攻击段或收手段，则不予更新；（只有攻击和收手才可能有技能盒）

若有碰撞盒处于攻击段开始后StartTime后，则对其进行一次目标寻找，根据目标确认碰撞盒位置；（但此时并不能对目标造成伤害）；

若有碰撞盒处于攻击段开始后的StartTime+DelayTime后，则开始对碰撞盒范围内的单位进行碰撞检测，并开始Update碰撞盒的角度旋转；（无位移Update）

不同技能的不同碰撞盒的不同间隔阶段，由于其攻击标识编号不同，可以对敌人造成多次伤害；同一攻击标识编号不能对同一敌人造成多次伤害。

3.选择技能盒目标单位List

根据SkillCollision表里的ColTarget参数配置，选择出场景中符合要求的潜在单位列表；（选取规则随后补充）

4.对所选取的潜在单位进行逐一过滤，规则如下：

a.若潜在单位的EEntityFlag.CanBeAttack标签为false，则该单位将被过滤掉；

b .若潜在单位已死亡，则该单位将被过滤掉；

c. 若潜在单位不在角色的仇恨列表里，则该单位将被过滤掉；

d. 若潜在单位已经被该碰撞盒攻击过（已保存过其攻击标识编号），则该单位将被过滤掉；

e. 当受击角色已被该技能碰撞盒攻击过，且受击角色的碰撞盒仍然在技能的碰撞盒范围中，则每次都检测技能碰撞盒的是否有新的拖拽盒已应用到受击角色中，如没有，则给受击角色添加技能拖拽盒；

f.若潜在单位不在技能碰撞盒范围内，则该单位将被过滤掉；

5.对筛选剩下的单位进行技能伤害结算

a.受击角色保存技能碰撞盒的攻击标识符编号，防止被再次伤害；

b.对受击角色应用技能伤害；

c.对受击角色应用技能伤害表现动画，如击飞等；

d.对受击角色首次应用技能碰撞盒里的拖拽盒表现效果；（后续应用在上面4的e中）

6.技能伤害后续处理

a.更新技能连击数；

b.应用使用技能的回馈增益效果：（加蓝）

获取施放技能单位的 蓝量获取增益系数 S：

获取技能的配置参数PowerGet；

技能施放者 回蓝 = （1+ S) * PowerGet；

7 通用技能阶段逻辑：

通用起手段Init：

a.应用技能起手段逻辑位移；

通用起手段Update：

通用起手段End：

通用攻击段Init：

a．应用技能攻击段逻辑位移；

b. 添加技能前置Buff和Aura；

获取技能配置SkillTemplate的PreBuffs和PreAuras参数,为单位添加对应Buff和光环.

c.应用技能特殊要求函数；

获取配置SkillTemplate的CanMove参数，如为0，则设定技能施放期间不能移动；

获取配置SkillTemplate的CanRotate参数，如为0，则设定技能施放期间不能转动；

d.开始计算CD

通用攻击段Update：

a.更新技能CD；

b.更新碰撞盒Update；

主要用于所有碰撞盒在各自Starttime后进行目标和位置初始，在StartTime+DelayTime后进行角度更新；

c.更新飞弹信息；

主要用于当所有飞弹在各自的startTime后进行发射；

通用攻击段End：

通用收手段Init：

a．应用技能收手段逻辑位移；

通用收手段Update：

a.更新技能CD；

b.更新碰撞盒Update；

主要用于所有碰撞盒在各自Starttime后进行目标和位置初始，在StartTime+DelayTime后进行角度更新；

c.更新飞弹信息；

主要用于当所有飞弹在各自的startTime后进行发射；

在此注意，配置的飞弹发射时间，必须在硬直结束前发射出去，硬直结束后就不允许发射飞弹了；

d.若收手段持续时间超过技能配置的AtkEndTime，则执行技能结束操作；

通用收手段End：

8.技能结束

c.恢复技能特殊要求函数；

无条件 恢复角色的移动属性，角色变为可以移动；

无条件 恢复角色的旋转属性，角色变为可以转动；
无条件 恢复角色的重力属性，角色变为受重力影响；