**X System高级教程**

**X System senior tutorial**

目录

[实例一 在应用了X系统的单位上添加更多自定义函式 2](#_Toc482017701)

[实例二 X 系统特质与单位自定义命令的结合 3](#_Toc482017702)

# 实例一 在应用了X系统的单位上添加更多自定义函式

由于应用X系统的单位必须加载X系统的自定义函式文件CustomCode.lua，因此我们需要解决的一个问题就是如何为这些单位添加更多自定义函式而又不影响X系统的运行。

首先，X系统要求单位在应用X系统时，CustomCode中<CustomGroup>一项填为与单位在Units列表中的名称一致，而Units列表中的名称又与单位的文件名称一致，可以用于区分各个单位，这就给我们接下来的工作提供了方便。

X系统当前并未使用<FunctionLoad>函式，想要添加该函式，只需在调用CustomCode.lua时将你的函式名填入该项即可。

至于另外三个函式，这里不推荐modder直接修改X系统的三个函式Create\_X、Update\_X和Destroy\_X，因为所有应用X系统的单位都在调用这三个函式，我们需要让添加的代码只针对特定舰船，这里，X系统在2.1.2版CustomCode.lua中提供了接口：X\_CustomFunctionCreate、X\_CustomFunctionUpdate和X\_CustomFunctionDestroy，它们都是table：

X\_CustomFunctionCreate={}

X\_CustomFunctionUpdate={}

X\_CustomFunctionDestroy={}

这里以X\_CustomFunctionCreate为例，其在Create\_X中被如下调用：

if(X\_CustomFunctionCreate[CustomGroup]~=nil)then

X\_CustomFunctionCreate[CustomGroup](CustomGroup, playerIndex, shipID)

end

因此，对于使用了X系统的单位，比如Hgn\_Scout，如果想要额外运行自己的自定义函式，比如一个额外的FunctionCreate，则只需在CustomCode.lua里定义函式如下：

X\_CustomFunctionCreate["Hgn\_Scout"]=function (CustomGroup, playerIndex, shipID)

……（你的代码）

end

该函式与FunctionCreate在参数、运行时机和使用方法上均完全相同，且只针对Hgn\_Scout单位，因此完全可以放心使用。

对于FunctionUpdate和FunctionDestroy，方法同FunctionCreate，这里不再赘述。

# 实例二 X 系统特质与单位自定义命令的结合

**Customcommand简介**

CustomCommand和CustomCode一样，是重置版提供的新的单位特性。在单位的ship文件中添加，格式如下（以Kus\_DroneFrigate为例）：

addAbility(NewShipType,"CustomCommand",1,"Drones",1,0,1000,200,0.25,2,0,"data:Ship/Kus\_DroneFrigate/Kus\_DroneFrigate.lua","Start\_DroneFrigate","Do\_DroneFrigate","Finish\_DroneFrigate","Kus\_DroneFrigate",1.15,2,1,0)

**NewShipType：**必须填NewShipType

**"CustomCommand"：**必须填"CustomCommand"

**1：**1代表单位默认拥有此能力，0代表单位默认不拥有该能力，但可通过其他方法获得该能力（如研究或子系统，具体获得方式参照其他能力，string应为“Custom”）

**"Drones"：**作用不明。

**1:** 作用不明。

**0:** 作用不明。

**1000:** 总能量（这里和单位的防御场能力的参数类似，自定义命令生效期间，会消耗“能量”能量耗尽后自定义命令停止执行）

200: 当能量小于或等于这个值时，可以手动解除自定义命令

0.25: 每0.1秒消耗的能量数。

**2:** 解除自定义命令时每0.1秒回复的能量数。

**0:** 执行自定义命令所需的最小剩余能量。

**"data:Ship/Kus\_DroneFrigate/Kus\_DroneFrigate.lua"：**调用的自定义命令代码文件路径。

**"Start\_DroneFrigate"：**自定义命令开始执行时调用的函式名称，函式在自定义命令代码文件中定义。

**"Do\_DroneFrigate"：**自定义命令执行过程中调用的函式名称，函式在自定义命令代码文件中定义。

**"Finish\_DroneFrigate"：**自定义命令停止执行时调用的函式名称，函式在自定义命令代码文件中定义。

**"Kus\_DroneFrigate"：**包含该单位的SobGroup（上述三个函式中的局域变量CustomGroup）的名称，通常同单位名称

**1.15：**自定义命令执行过程中调用函式的频率，单位s

**2：**决定自定义命令的执行按钮调用哪个图标（在ui\newui\taskbar\tb\_commandpanel.lua中定义）

**1：**作用不明。

**0：**自定义命令执行完毕后舰船是否会自毁。

现在以FX：Galaxy中的光之云母为例，她有如下特质：

{

"OneselfChangePower",

"Other",

{

{

{"IsShipIdioStateKeyOn",1,},

{"IsControlledBy",1},

},

{

{"IsAttackingFamily",0,"SmallCapitalShip,BigCapitalShip,Mothership",},

{"IsControlledBy",-1},

},

},

{},

{},

{{"ResourceStart",},{},16,0},

{{"ResourceDo",},{},4,0},

{{"ResourceEnd",},{},12,0},

{1,1,},

{"Weapon\_PhasedCannon",},

{"",},

},

{

"OneselfRepair",

"Add",

{

{

{"IsCharacteristicActive",-1,},

},

},

{},

{},

{{},{},0,0},

{{},{},0,0},

{{},{},0,0},

{1,1,},

0.0064,

},

{

"OneselfSpeedChange",

"Multiplication",

{

{

{"IsCharacteristicActive",-1,},

},

},

{},

{},

{{},{},0,0},

{{},{},0,0},

{{},{},0,0},

{1,1,},

1.5,

},

转换为人类语言就是，光之云母被玩家控制且ID=1的状态Key开启时，或者被AI控制且正在攻击AttackFamily为以下几种之一时:SmallCapitalShip,BigCapitalShip,Mothership，则Weapon\_PhasedCannon武器可用（动画部分略），反之，该武器不可用，但是获得每秒0.64%的生命恢复特质和50%的加速特质。在游戏中，这一系列特质表现为光之云母在进攻模式和防御模式之间的状态切换，切换到进攻模式时，光之云母经过复杂的动画展开后，能够使用其主炮，而切换到防御模式后，经过收拢动画，光之云母主炮关闭但是获得生命恢复和加速能力。

这里，光之云母被AI控制时的条件不存在问题，但是被玩家控制时，需要ID=1的状态Key开启才能切换到进攻模式。而该单位并没有控制状态Key的三个特质，这里就是通过单位自定义命令实现的。

首先来看光之云母的单位自定义命令代码：

addAbility(NewShipType,"CustomCommand",1,"Missile",1,0,1, 1, 1, 1,1,"data:leveldata/multiplayer/resdata/CustomCode\_Switch.lua","Start\_X\_Switch","Do\_X\_Switch","Finish\_X\_Switch","kad\_mothershiplight",1.5,1,1,0)

该命令调用了X系统提供的范例文件Customcode\_Switch.lua，里面有Start\_X\_Switch,Do\_X\_Switch,Finish\_X\_Switch三个函式:

function Start\_X\_Switch(CustomGroup, playerIndex, shipID)

if(xGetShipIdioStateKey(shipID,1)=="Off")then

xSetShipIdioStateKey(shipID,1,"On")

else

xSetShipIdioStateKey(shipID,1,"Off")

end

end

function Do\_X\_Switch(CustomGroup, playerIndex, shipID)

end

function Finish\_X\_Switch(CustomGroup, playerIndex, shipID)

end

于是我们明白了问题所在，Start\_X\_Switch函式是关键，该函式使用了X系统提供的公共函数xGetShipIdioStateKey和xSetShipIdioStateKey，能够切换光之云母的状态Key，也就是说，当玩家通过热键或界面上的按钮手动执行了光之云母的状态切换特质时，CustomCode\_Switch.lua中的Start\_X\_Switch函式会被调用，而后，光之云母ID=1的状态Key的状态就会改变，如果之前处于关闭状态，则打开，激活武器切换特质，关闭生命回复和加速特质，进入进攻模式，此后，该Key保持打开状态，云母即停留在主炮激活的状态，而云母的自定义命令持续时间非常短暂，几乎没有持续和冷却的时间，因此自定义命令按钮立刻就能再次被点击，于是当玩家再次按下该按钮，光之云母状态Key即关闭, 进入防御模式。这就将该单位的自定义命令按钮改造成了状态切换按钮。

CustomCode\_Switch.lua适用于绝大部分类似特质，如卡德仕支援座舱。通过这种方法可以实现了X System的单位特质与单位的自定义命令相结合。此外还有另一个范例CustomCode\_Missile.lua，适用于拥有持续和冷却时间的技能型特质，如图拉尼攻击运载舰、守护者的金钟罩等。该文件使单位的状态Key与其自定义命令保持同步，激活期间，状态Key打开，关闭或冷却期间，状态Key保持关闭。