# 新木偶框架规范

## 基础API

导入类：

saltsheep.ssl.api.neo.AnimationNode

创建动画节点：

var node = new AnimationNode(long durationTime);

创建一个AnimationNode，传入参数为动画播放时间，单位为毫秒

播放动画节点内容：

node.play(EntityNPCInterface npc)

应传入有木偶职业的NPC的MC实体

*注：新木偶框架支持同时有多个动画，它会按顺序依次播放*

设定动画内容：

设置动画节点的终点角度：

node.setRotation(int part, float x, float y, float z)

返回node本身，用于链式调用

node.setRotation(int part, float x, float y, float z, TaskPlay.PlayMode mode)

设置角度的升级版，可以设定动画播放的模式，

模式封装在saltsheep.ssl.puppet.handler.tasks.TaskPlay.PlayMode中，有如下：

NORMAL,

STF\_SMALL,FTS\_SMALL,

STF\_NORMAL,FTS\_NORMAL,

STF\_LARGE,FTS\_LARGE

其中NORMAL为匀速旋转至终点，

STF为先慢后快，FTS为先快后慢，后缀代表了速度的最大值程度

(SMALL:1.57,NORMAL:2,LARGE:3)

该方法同样返回node本身，用于链式调用

立刻结束剩余所有动画：

node.clear()

会使未播放完的动画立即结束，并且到达其最终角度，返回node本身

*注：如果你有兴趣，可以看看底层实现，事实上，clear是预设的一种动画任务，是分配到指定部位上的。由此，尽管你可以在AnimationNode中使用clear后再设置角度，这会有效，但我不建议这么做。为了保证动画按节点播放，未设置旋转的部位使用Delay替代，而Clear后所有部位未空，不存在Delay。如果你需要暂停动画的操作，我建议先clear一次，再用另一个动画节点。*

清除动画并回归原始状态（全180°）：

node.reset()

是clear的升级版，在结束剩余动画之后，它会将角度还原，返回node本身

Tips：启用/关闭指定部位木偶动画是否可用，请看二级API

## 二级API

导入类：

saltsheep.ssl.api.neo.AnimationHandler

*注1：我比较推荐在创建动画节点后，使用AnimationHandler来操作，这样的可读性会稍好，并且AnimationHandler包装了clear和reset的功能，相对更加方便。此外，AnimationHandler支持传入ICustomNpc，AnimationNode并不支持——因为我的初始设计是使用AnimationHandler来调用动画，AnimationNode更接近实现层*

*注2：事实上，AnimationNode已经完全够用，AnimationHandler基本基于AnimationNode，部分内容（部位启用）来自旧木偶框架，其设计目的在于平滑地过渡旧木偶框架脚本，当用新AnimationHandler传入旧Animation时，会将其转换为AnimationNode。*

所有方法皆为静态方法

AnimationHandler.addAnimation(npc,ani)

npc传入ICustomNpc/EntityNPCInterface

ani传入Animation(旧框架动画)/AnimationNode

等价于ani.play(npc)

AnimationHandler.stopAnimation(npc)

npc传入ICustomNpc/EntityNPCInterface

等价于new AnimationNode(0).clear().play(npc)

AnimationHandler.resetAnimation(npc)

npc传入ICustomNpc/EntityNPCInterface

等价于new AnimationNode(0).reset().play(npc)

AnimationHandler.setPartUseAnimation(npc, int part, boolean isUseAnimation)

npc传入ICustomNpc/EntityNPCInterface

设置是否启用指定部位的木偶动画，效果等同于旧木偶框架（直接调用了旧框架的方法），或者可以理解为，原生的木偶部位可用设置后立即更新到客户端。

AnimationHandler.translate(Animation ani)

返回AnimationNode，将旧框架Animation转换为新框架AnimationNode

## 包装API

导入类：

saltsheep.ssl.api.neo.AnimationPartWrapper

内容：HEAR,LARM,RARM,BODY,LLEG,RLEG (全为int类型静态变量)

对应部位的包装，用于提高代码可读性，前置L代表左，R代表右，ARM手臂，LEG腿

## 底层实现API（并非必学！）

详细请见包saltsheep.ssl.puppet

此部分内容涉及较复杂的开发，我将默认使用者拥有一定的基础。

实现一 ASM与逻辑：

使用ASM修改了JobPuppet的三个获取角度方法，具体请看

saltsheep.ssl.puppet.asm.Trans

saltsheep.ssl.puppet.handler.NeoPuppetHandler

Trans负责ASM内容实现

NeoPuppetHandler负责角度的逻辑实现

实现二 部位列表化，任务化：

请见saltsheep.ssl.puppet.handler.JobPuppetSSLData

每个JobPuppet都有一个JobPuppetSSLData

在获取角度时，如果指定部位有数据，该部位会启用新木偶框架，否则为原生木偶

为了支持某些潜在的，支持了更多部位的NPC附属mod，我没有写死部位种类及数量

有数据的部位将有一个对应的RotationController

RotationController有如下四个属性：

xStart,yStart,zStart,tasks

前三者记录最后的旋转位置，tasks记录收到的动画任务

当有动画任务，会通过动画任务获取角度；无动画任务，会得到最后记录的旋转位置

实现三 任务的实现：

三个关键类：

saltsheep.ssl.puppet.handler.JobPuppetSSLData.IAnimationTask

saltsheep.ssl.puppet.handler.JobPuppetSSLData.IAnimationTaskLoader

saltsheep.ssl.puppet.handler.tasks.PuppetTaskLoaders

IAnimationTask决定了任务的具体实现

IAnimationTaskLoader负责在服务端-客户端的序列化

PuppetTasksLoaders负责登记和托管所有IAnimationTaskLoader

目前实现的IAnimationTask有：

Clear,Delay,Play,Rest

你可以考虑自己扩展（鉴于双端性质，必须要使用模组/插件开发，双端脚本）

实现四 数据包缓存：

见saltsheep.ssl.puppet.network.SPacketAnimation

如果一个NPC有过任何新木偶框架的操作，都会为其添加一个缓存数据包

这个缓存数据包将会在游戏刻的末尾发送给客户端，这是为了保证任务顺序的一致

在数据包发送之前，可以无限量地添加新任务给指定部位（详见SPacketAnimation）