

Open And Close (OAC Devkit)

开和关开发包文档

Version: 3.0

[PDF Document](#)

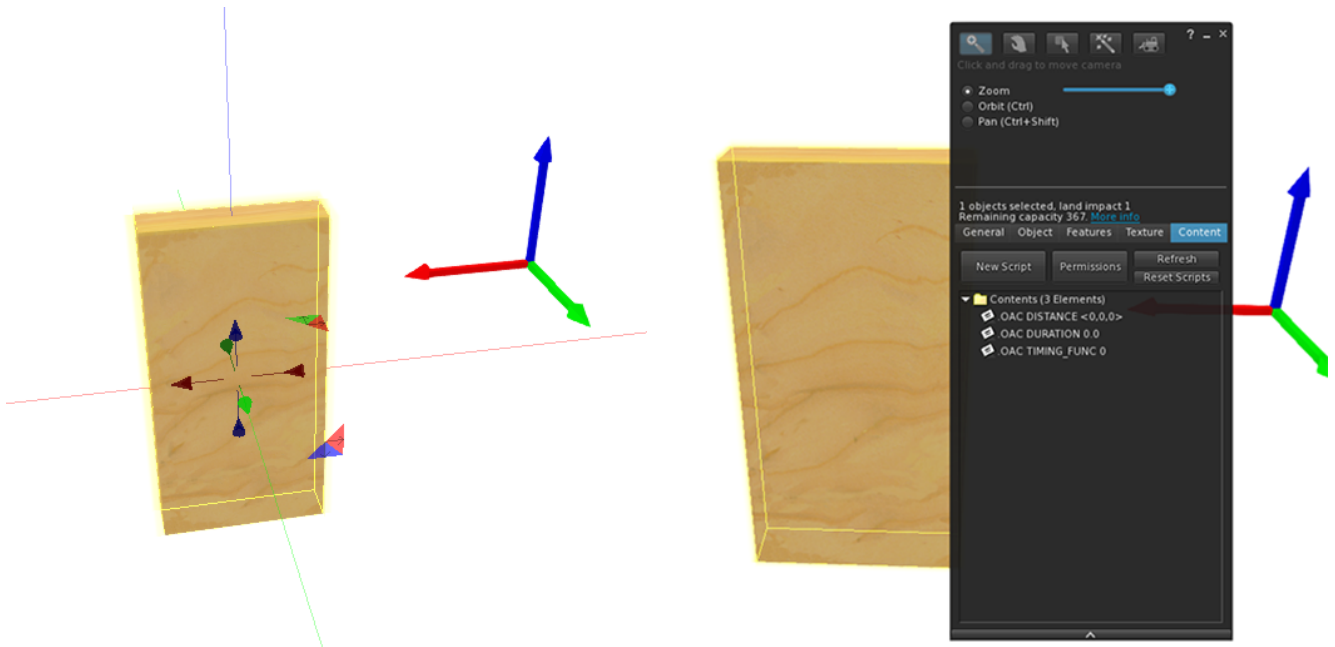
功能

- 平滑的效果，平滑的视觉。
- 灵活的配置和组合。
- 变形过程中，可以随时改变方向。

快速开始。按照这个，一步一步

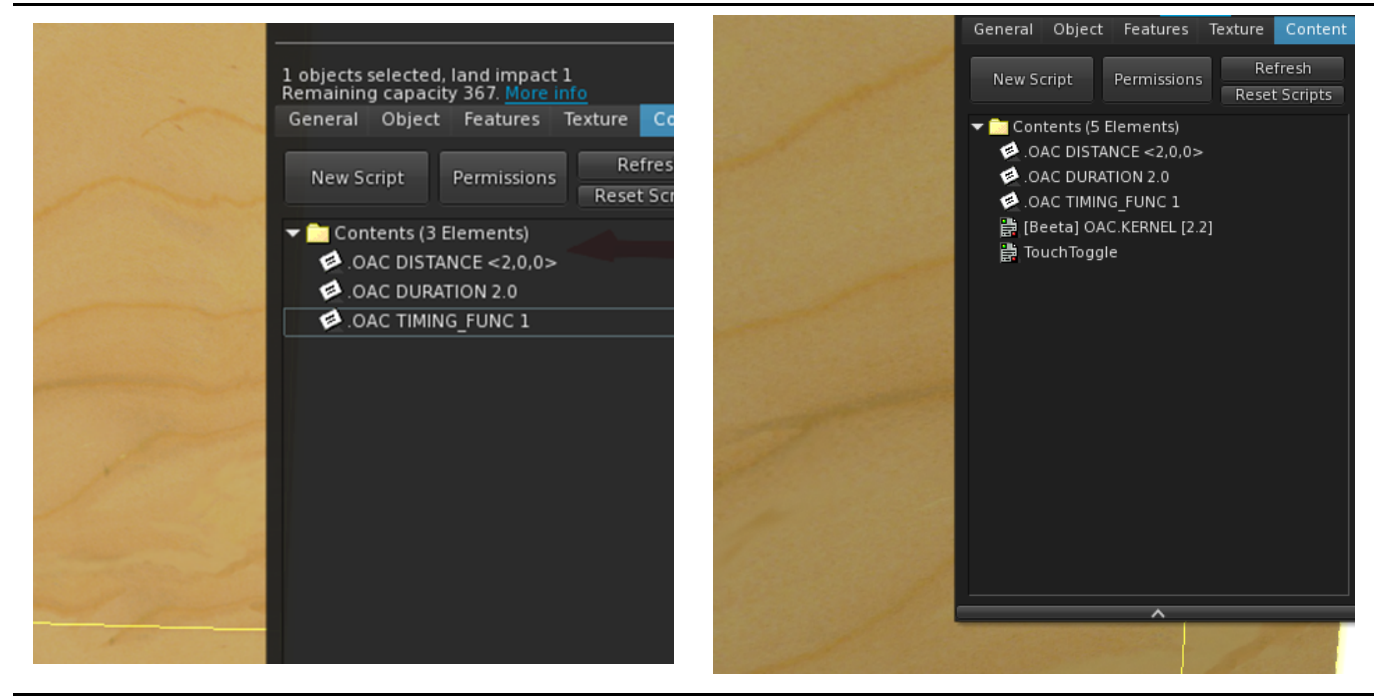
1. 准备您的物品
2. 根据自己的需要，选择以".OAC"开头的配置文件，修改其参数，拖入清单。
3. 将名为OAC.KERNEL 的主脚本拖到清单中。
4. 选择您需要的触发脚本，将其拖放到对象中。"Extra"中已经为您预设了一些触发脚本。当然，您可以根据需要自定义它们。
5. 完成。

做一个简单的推拉门



创建一个盒子，像门一样

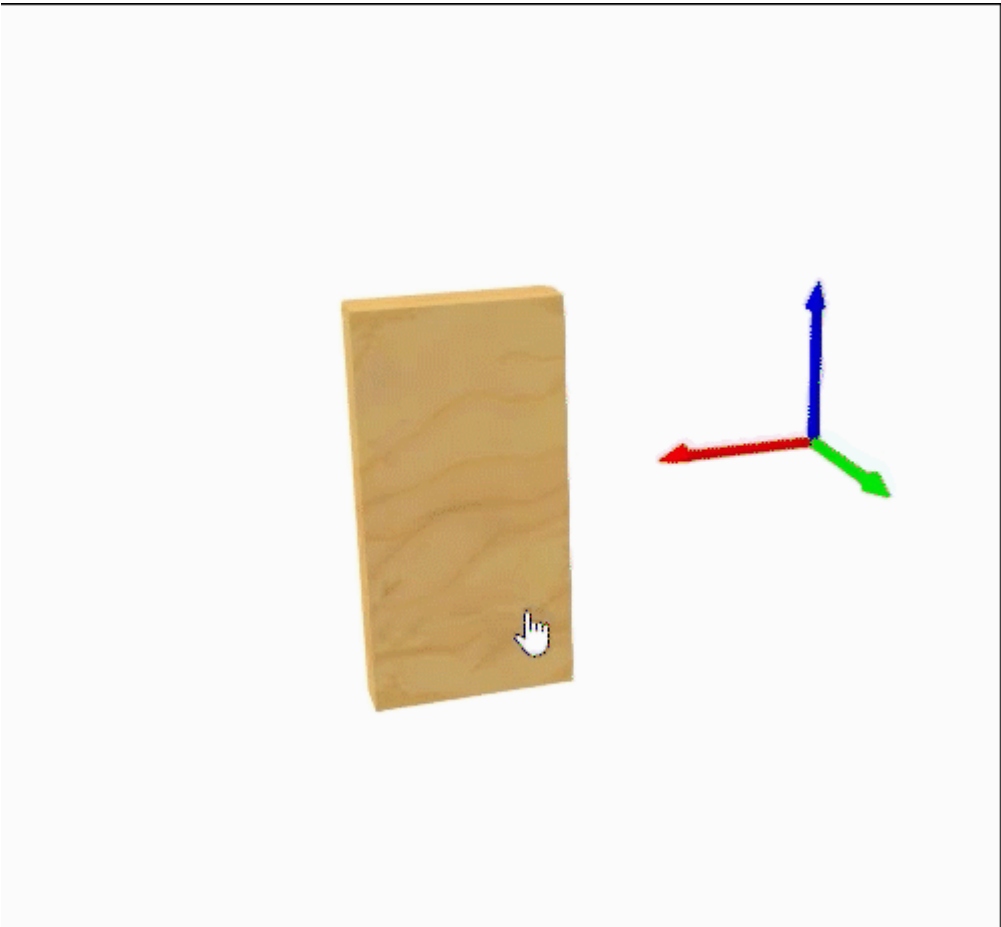
选择您需要功能的配置文件，将它们拖放到目录中



改变参数
X方向移动2米
持续时间2秒
使用 ease-in-out 效果功能

拖放脚本

点击查看效果



更详细的例子，在"Example"目录中，放出来进行测试和编辑

包含脚本

名称	描述
OAC.KERNEL	(必要的) 主脚本

Extra

名称	描述
TouchToggle	使 prim 可点击，触发并切换往复运动，它只会触发当前 prim(LINK_THIS)。
TouchToggleSync	使 prim 可点击，触发并切换往复运动，它会触发(LINK_SET)中所有prim，通常用在根prim。
AutoClose 30s	打开30秒后自动关闭。
AutoToggle after end 20s	转换结束后，等待20秒切换状态，循环。
AgentSensorOpen	附近有人时打开。
AgentSensorToggle	附近有人时打开，无人时关闭。
SoundTrigger	运行过程中播放声音，此脚本预设为电动门，可任意更换。
TouchToggleQueue	(≥ 3.0) 使 prim 可点击，触发Queue模式的运动，它只会触发当前 prim(LINK_THIS)。
TouchToggleSuncQueue	(≥ 3.0) 使 prim 可点击，触发Queue模式的运动，它会触发(LINK_SET)中所有 prim，通常用在根prim。

配置

一个notecard(记事卡)代表一个配置字段，拖拽到内容栏，编辑它名字以修改参数。

格式: .OAC {关键字} {值}

关键字	类型	取值	默认	描述	版本
DURATION	float	任何	0.0	时长，如果小于0.1，则视为0.0，0.0表示没有运动过程，瞬间完成	1.7
DISTANCE	vector	任何	<0.0,0.0,0.0>	距离，移动变化	1.7
ROTATION	vector	任何	<0.0,0.0,0.0>	旋转，旋转变化，这个向量的含义是<ROLL, PITCH, YAW>。 * 旋转总是相对于prim的局部(local)方向向量。	1.8
SCALE	vector	大于 <0.0,0.0,0.0>	<1.0,1.0,1.0>	缩放，缩放变化，不可出现负值，如果等于ZERO_VECTOR (<0.0,0.0,0.0>)，则视为无效的	3.0

关键字	类型	取值	默认	描述	版本
ORIGIN	integer	0/1/2	0	参照物，见下方特别说明	2.0
TIMING_FUNC	integer	0/1/2/3	0	过渡效果，见下方特别说明	2.0
QUEUE	string			Queue模式，相见下文	3.0

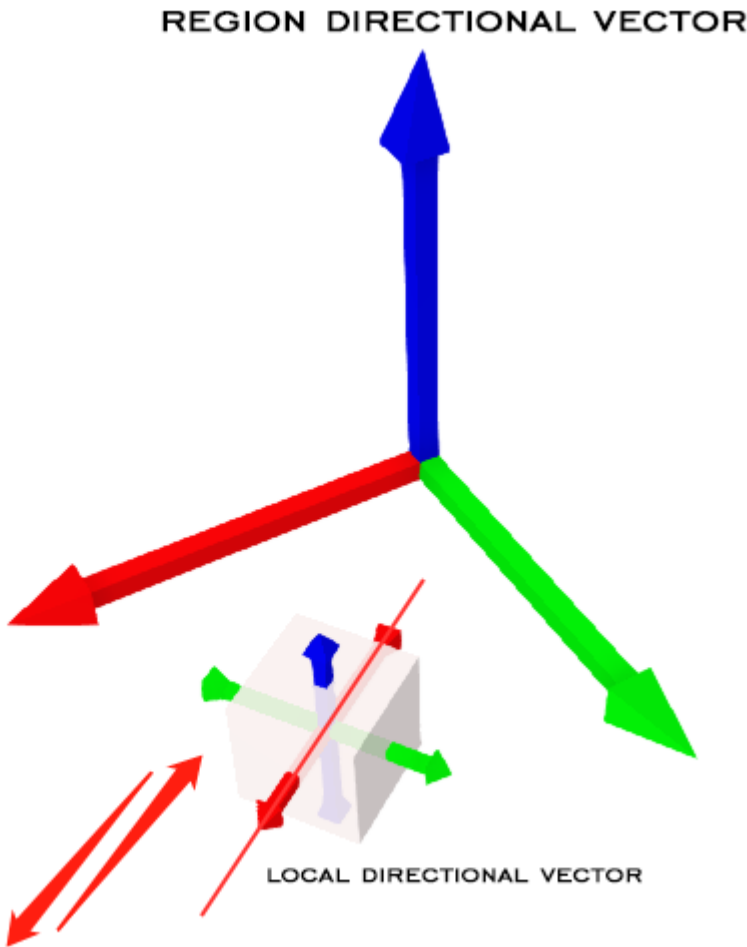
关于 参照物 ORIGIN

局部(local) (0)

运动方向将参考局部(local)方向向量。

例子:

```
.OAC DISTANCE <1.0, 0.0, 0.0>
.OAC ORIGIN 0
```

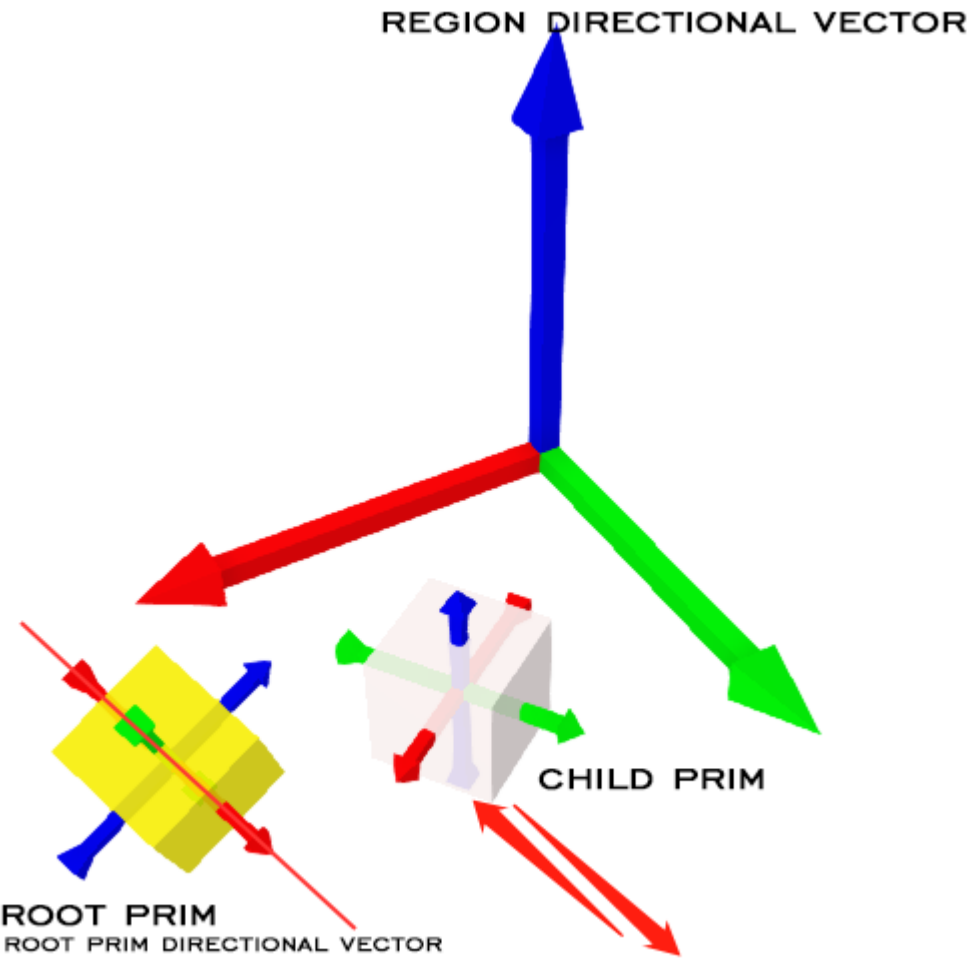


根prim(root) (1)

运动方向将参考根prim的全局方向向量。

例子:

```
.OAC DISTANCE <1.0, 0.0, 0.0>
.OAC ORIGIN 1
```



它仅适用于链接集中的子 prim。当对象是根 prim 或者它是一个独立的 prim 时，视为全局。

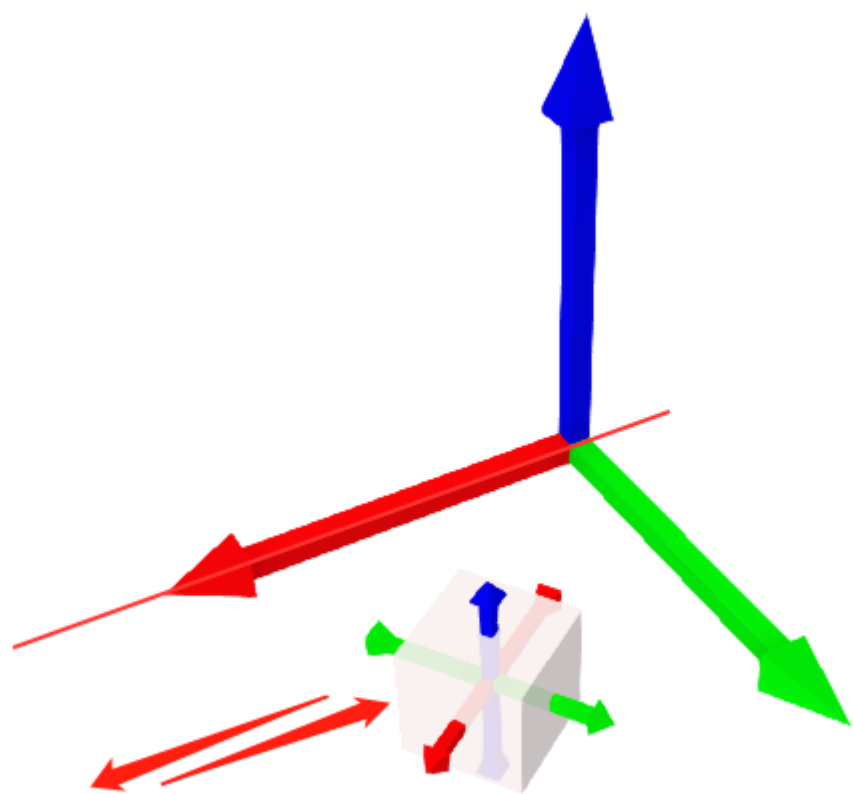
全局(world) (2)

转换将参考全局(world)方向向量。

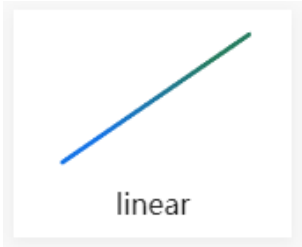
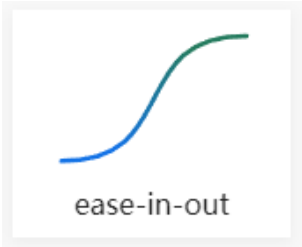

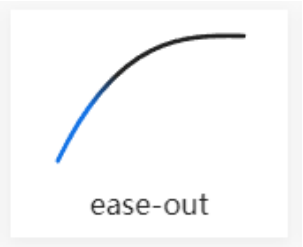
例子:

```
.OAC DISTANCE <1.0, 0.0, 0.0>
.OAC ORIGIN 2
```

REGION DIRECTIONAL VECTOR



关于 过度效果 TIMING_FUNC

0: linear 线性	1: ease-in-out 缓入/出	2: ease-in 缓入	3: ease-out 缓出
 linear	 ease-in-out	 ease-in	 ease-out
.OAC TIMING_FUNC 0	.OAC TIMING_FUNC 1	.OAC TIMING_FUNC 2	.OAC TIMING_FUNC 3

Queue 模式

在 3.0 版本中新增Queue模式，它可以连续演绎多个变化过程（正向、反向），并且延续了任意时间点随时切换方向的特性。

```
.OAC QUEUE {编号}/{时长}/{参照}/{时间函数}/{距离}/{旋转}/{缩放}
```

是的，它将以前所支持的参数写在一行，并赋予给QUEUE，然后，您可以添加多个QUEUE。

{编号}代表了QUEUE顺序，在PRIM的内容里，文件是按照文件名升序顺序排列的，所以只要能保证顺序的正确，编号可以随意指定，无论是 1234... 或者 ABCD...。

如果两个QUEUE中需要等待，可以加入一个只带有时长的QUEUE，像下面这样：

```
.OAC QUEUE 1/5.0///<10.0,0.0,0.0>//  
.OAC QUEUE 2/2.0/////   
.OAC QUEUE 3/5.0///<0.0,10.0,0.0>//
```

调用

在指令后面增加 "|1"

```
llMessageLinked(LINK_SET, 802840, "OPEN|1", "");  
llMessageLinked(LINK_SET, 802840, "CLOSE|1", "");  
llMessageLinked(LINK_SET, 802840, "TOGGLE|1", "");
```

本地消息接口

本地控制与数据提交

Num: **802840**

开/正向变换

positive movement

```
llMessageLinked(LINK_SET, 802840, "OPEN", "");
```

关/反向变换

reverse movement

```
llMessageLinked(LINK_SET, 802840, "CLOSE", "");
```

正反向切换

Switch the current direction of movement

```
llMessageLinked(LINK_SET, 802840, "TOGGLE", "");
```

本地事件广播

Num: **802841**

变换开始

发送至: LINK_SET

```
TRANSFORM_STARTED|{方向}
```

方向:

- 1: 开, 正向变换
- -1: 关, 逆向变换

变换结束

发送至: LINK_SET

```
TRANSFORM_FINISHED|{方向}
```

方向:

- 1: open, positive movement
- -1: close, reverse movement