## 概述

### 插件介绍

核心插件：

◆Drill\_CoreOfBallistics 系统 - 弹道核心

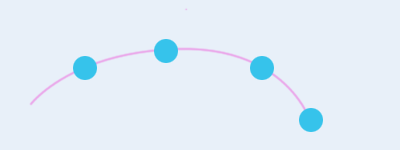
子插件：

◆Drill\_CoreOfGaugeMeter 系统 - 参数条核心

◆Drill\_CoreOfShatterEffect 系统 - 方块粉碎核心

这里的弹道与实际的弹道学有一些区别，

弹道定义：能够描述一群粒子/碎片/子弹运动的轨迹，是程序**数学计算**后的结果数据集。



### 插件关系

弹道核心属于数学计算，服务于其他的核心，关系如下：



## 弹道模式

核心提供了下列弹道的移动模式。需要注意的是，相关子插件会根据某些具体的情况，提供部分模式的设置，但不一定会提供所有模式的支持。

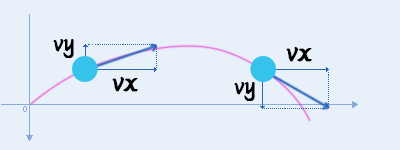
### 直角坐标模式

直角坐标模式通过 x速度 + y速度 控制轨迹。时间、速度已知，路程未知。

**X速度(vx)**：正数向右，负数向左。单位 像素/帧。见 [速度属性](#_速度属性) 。

**Y速度(vy)**：正数向下，负数向上。单位 像素/帧。见 [速度属性](#_速度属性) 。

（注意，y速度正数向下，是反的。）



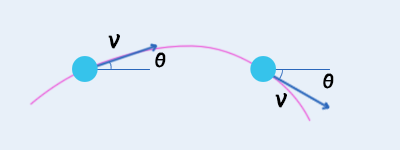
### 极坐标模式

极坐标模式通过 速度+方向 控制轨迹。时间、速度已知，路程未知。

**速度(v)**：正数沿着方向移动，负数反向移动。单位 像素/帧。见 [速度属性](#_速度属性) 。

**方向(θ)**：0朝向右，90朝下，-90朝上，顺时针规律。单位 角度。见 [方向属性](#_方向属性) 。

（可以写超过360的数。390度与30度的方向一样。）



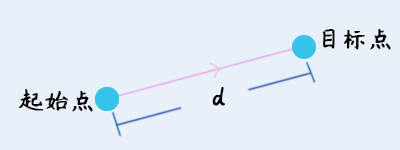
### 两点式

两点式通过 固定模式+目标点 控制轨迹。时间、路程已知，速度未知。

**目标点：**分为绝对坐标与相对坐标，用于计算当前物体到目标物体的路程。

**固定模式：**由于时间和路程是固定量，速度是可控制变量。根据速度的不同变化方式，可分成多种模式。见 [固定模式属性](#_固定模式属性) 。

（两点式设置与 窗口的移动动画设置 相似）



## 弹道属性

### 速度属性

**1）初速度**

速度1，表示 1像素/帧。 速度1.5，表示 1.5像素/帧。

速度可以为负数，表示反方向移动。

只有初速度时，表示匀速运动。

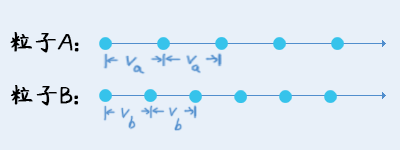


**2）随机波动量**

在粒子的速度的基础上，添加/减少的随机速度量。

比如，初速度为10，波动量为8，那么速度为：6~14之间。（10-4 ~ 10+4）

初速度+波动量 仍然为匀速运动。

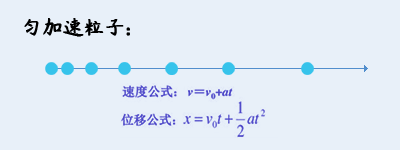


**3）加速度**

粒子在运动过程中，速度逐渐改变的加速度值。

极坐标模式中：无论加速度如何变化，都**不会**影响运动方向。

直角坐标模式中：如果vy的加速度化和vx的加速度**不一样**，那么就会间接改变这个粒子的实际方向。比如，vx匀速+vy匀加速 = 抛物线。



**4）速度最大/最小值**

设置限制后，无论速度如何变化，都会保持在最大速度和最小速度之内。

比如，一个不断减速的粒子，如果设置了最小速度0，则会停下来。否则粒子速度将会越来越小，变成负数，朝反方向开始移动。

**5）路程计算公式**

见后面的章节”路程计算公式”。

### 方向属性

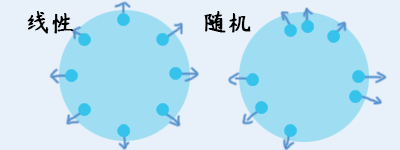
**1）方向单位**

主要方向单位为 角度 ，部分设置里面可能会使用弧度，360（角度）= 2π（弧度）。

如果特殊情况要求输入弧度时，可用3.14代替π。

**2）四周扩散**

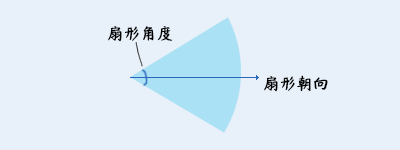
四周扩散，即以中心点为基准，方向随机。

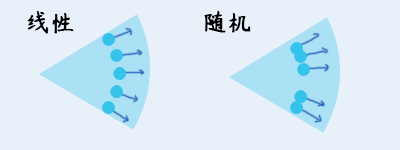


**3）扇形范围**

图中的朝向的角度为0，90度朝下，-90朝上，顺时针规则。

扇形角度为360时 = 四周扩散





**4）方向计算公式**

见后面的章节”方向计算公式”。

### 固定模式属性

（待填坑，未写完）

**1) 不移动**

**2) 匀速移动**

**3) 增减速移动**

**4) 弹性移动**

## 数学公式

### 路程计算公式

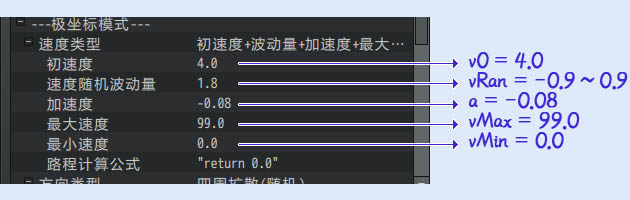
#### 参数与变量

**速度、加速度等，都是通过路程计算公式配置的**。

无论是极坐标模式，还是直角坐标模式，路程都决定了粒子**最终所在的位置**。

公式中提供了id,time,v0,vRan,a,vMax,vMin变量。

**其中，只有id、time是游戏中变化的量，vRan是变化的随机值，其它的都是配置参数的固定值**。你可以直接写死这些量。



你如果有更多轨迹的设计想法，不妨可以了解后面的原理。

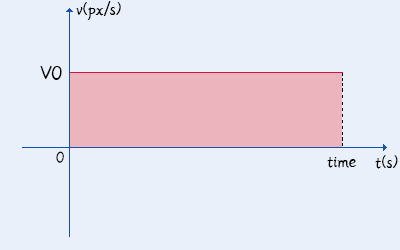
#### 公式：只初速度

众所周知，r = v\*t，路程=时间\*速度。匀速运动的情况最简单，长方形面积就是。

v0为初速度，time为时间。return表示返回结果。

则公式为： **return v0\*time**

（注意，只有time是游戏中变化的量，v0就是配置中的数值）



只初速度的v-t图

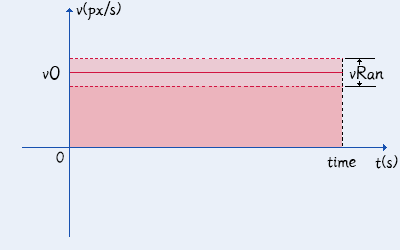
#### 公式：初速度+波动量

波动量是指，初速度有一定的随机性，可以在上下进行浮动。

v0为初速度，vRan为波动量，time为时间。（vRan在使用时会自动赋随机值。）

则公式为： **return (v0+vRan) \* time**

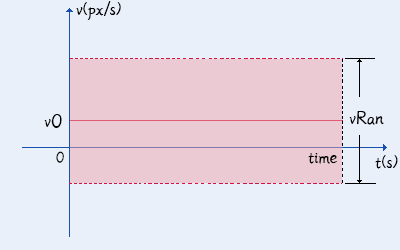
（注意，只有time是游戏中变化的量，vRan是随机值，v0就是配置中的数值）



初速度+波动量的v-t图

需要注意的是，如果vRan波动量比v0初速度还大，那么就会出现速度为负值的情况。

速度为负，则会朝相反的方向移动。



初速度+波动量的v-t图

#### 公式：初速度+波动量+加速度

匀加速运动的公式为：r=vt+½at²，也就是图中的长方形和三角形的和。

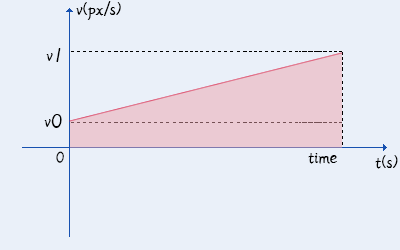
v0为初速度，a为加速度，time为时间。v1=v0+a\*time。

则公式为： **return v0\*time+0.5\*a\*time\*time**

加上波动量的公式为： **return (v0+vRan)\*time+0.5\*a\*time\*time**

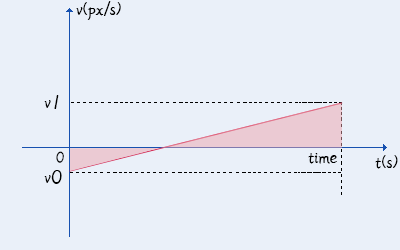
（如果你熟悉js的一些数学函数用法，也可以使用 Math.pow(time,2) 表示时间的平方 ）

（注意，只有time是游戏中变化的量，vRan是随机值，v0、a就是配置中的数值）



初速度+波动量+加速度的v-t图

如果出现速度为负数情况，公式依然成立。



初速度+波动量+加速度的v-t图

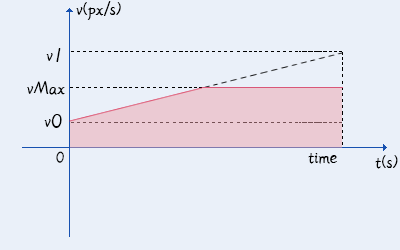
#### 公式：初速度+波动量+加速度+最大最小

最大速度和最小速度是一个十分特殊的情况。

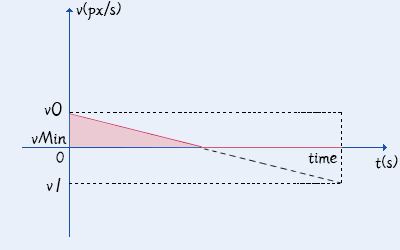
它会将情况分成多段。

如果速度超出了最大速度，那么就**减去**vMax多出的面积。

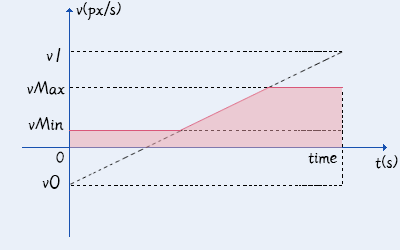
如果速度低于了最小速度，那么就**加上**vMin多出的面积。



最大速度切割v-t图



最小速度切割v-t图



最大最小速度组合切割的v-t图

v0为初速度，a为加速度，time为时间，vMax为最大速度，vMin为最小速度。

v1=v0+a\*time。

公式为：

**var v1 = (v0+vRan) + a\*time;**

**var d = (v0+vRan)\*time+0.5\*a\*time\*time;**

**if( v1 >= vMax ){**

**var m\_v = v1-vMax;**

**var m\_t = (v1-vMax)/a;**

**d = d - m\_v\*m\_t+0.5\*a\*m\_t\*m\_t;**

**}**

**if( v1 <= vMin ){**

**var m\_v = v1-vMin;**

**var m\_t = (v1-vMin)/a;**

**d = d - m\_v\*m\_t+0.5\*a\*m\_t\*m\_t;**

**}**

**return d**

（注意，只有time是游戏中变化的量，vRan是随机值，v0、a、vMax、vMin就是配置中的数值）

### 方向计算公式

暂无，留坑

### 透明度计算公式

暂无，留坑