

part 因为作者学业繁忙，这个资料有很多未完成的地方，日后会尽快补齐

简 单 火 箭 2

v i z z y 从 入 门 到 入 坟

Simplerocket2

游戏版本 V:0.9.918

作者:

所罗门老狗 普朗克的钟 カチューシャ
V3.7.15

目录

壹 基本介绍	
基本结构	03
程序的运行	03
更多	03
贰 翻译及注释	
Program flow (程序流程)	04
operators (运算符)	06
Craft Instructions (工艺说明)	10
Craft Information (作品信息)	13
Events (事件)	19
variables (变量)	00
lists (列表)	00
Custom Expressions (自定义表达式)	00
Custom Instructions (自定义指令)	00
叁 程序实例	
用 pid 使火箭悬停	00
肆 其他有关内容	
Vizzy 中的符号	00
基本概念	00
刚体力学	00
伍 在 xml 中编写	
程序的储存和信息	00
事件类	00
变量类	00
陆 尾页	

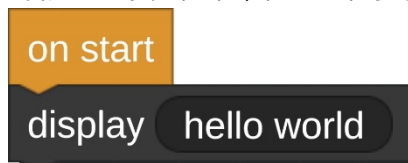
壹 基本介绍

Vizzy 是一个可视化的编程系统, 相较于其他编程引擎有跟高的可读性, 但是效率欠佳, 总的来讲凑合可用, 而且对于这个游戏里来讲已经完全够用了。

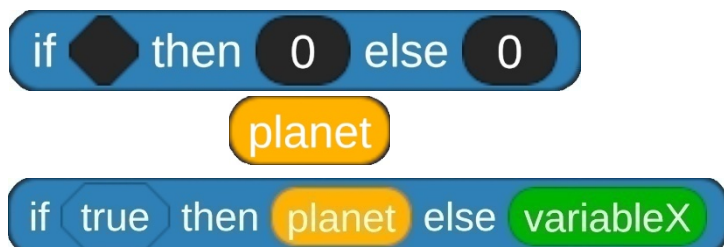
Vizzy 不仅能写在芯片里, 同样也可以写在任意一个零件里面。vizzy 只会在玩家操控的飞船的物理距离内运行, 如果太远会被优化掉, 不用担心, 靠近了又会继续运行。

一：基本结构

Vizzy 跟其他编程引擎类似, 自上而下依次运行, 只是更加直观和便利, 它没有严格点的格式要求, 在菜单栏里有很多模块, 可以被拖出来链接到对应的地方。



效果如图此类型的模块都可以上下串联起来, 并依次运行



这些圆圆的模块可以被放在这些黑黑的洞里, 只要概念没有错都是可以运行的
其次 vizzy 没有什么严格的格式要求可以放心食用, 非常适合程序小白图一乐

二：程序的运行

貳 翻译及注释

在 vizzy 中可以被执行的程序和数据被称为模块，可以拖动某些，模块使其从上到下排列，系统会根据排列顺序以一定速度依次对模块做出相应的反馈，并进入下一个模块直到结束，以此来完成用户所编辑的 vizzy 程序

一：program flow（程序流程）

此目录下的程序可以连接原本独立的程序片段而形成一个连续完整的程序、，更好的使用能帮你完成更大更复杂的程序组。

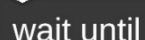
1: wait () seconds （等待 () 秒）

A Scratch 'wait' block with the value '1' and the unit 'seconds'.

等待一秒并非现实世界的一秒，这存在一点误差，受是设备刷新帧率影响而产生的，并不大但不可忽视，在时空扭矩中会发生极大错误

如图，在 wait () seconds 上面的程序片段运行完毕后等待 () 秒，再开始运行下一个程序片段，如果上面的程序没有运行则无效

2: wait until <> （等待直到 <>）

A Scratch 'wait until' block with a diamond-shaped condition slot.

等待直到菱形方框内执行或达成则开始运下一行程序

例：wait until <AGL≤5000> 意思就是等待直到距地面高度小于 5000m 运行下一步

注：如果未达成 wait until <> 里面的条件后面的程序是无法执行的，即使后面的判定是达成的

3: repeat () （重复 () 次）

A Scratch 'repeat' block with the value '10'.

可以在 () 里填入需要重复的次数，此程序所包含的程序将会运行 () 次，重复完成后会自动跳出循环

4: while <> 在 <> 期间

A Scratch 'while' block with a diamond-shaped condition slot.

当 <> 为真命题时无限循环运行所包含的程序，直到 <> 内不再为真命题才会跳出，如 <> 为否命题则直接跳过此程序和所有包含的子程序直接运行下面的程序

英文本意是在……期间，while 会一直无限循环被包含的模块，例如在“高度 < 500 期间”，那么在高度小于 500 的时候会一直依次循环所包含的模块，直到高度大于 500 才会跳出循环并运行以下的

注:while <true> 可以实现无条件无限循环，强行打断循环需要 brake

5: for (i) from (A) to (B) dy (C) （通过 (C) 从 (A) 设置 (i) 到 (B)）

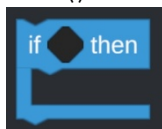
A Scratch 'for' block with 'i' as the loop variable, '1' as the start, '10' as the end, and '1' as the step.

此程序将通过 by 后面的数字或公式运算来 (i)，每一次运算会附带此模块包涵的模块依次执行，包涵的模块依次运行完成以后进行下一次运算，直到 (i) 等于或超过 to 后面的数，多次重复计算，完成后会自动跳出循环执行以下的模块。(i) 不会超过 to 后面的数，即使最后一次计算会超出 to 后面的数，(i) 的度数只会停留在 to

注:数值不能存在明显错误，如图 (i) 的初始值为 1 通过-1 永远无法达到 10，程序不会无限循环，而是出现错误停滞

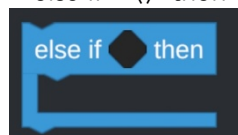
类似于 C++ 中的 for 循环

6: if <> then (如果 <> 那么)



与 while 类似，只有 <> 为真命题时才运行被包涵的模块，若为假命题则直接跳过以及被包涵的模块直接运行以下的模块，和 while 不同的是此模块只会运行一次，判断 <> 条件达成后运行包涵的模块，完成后自动跳出循环运行以下模块，不会再次判断 <>

7: else if <> then (除非 <> 否则)



与 if <> then 性质相同且相反，当满足 <> 条件时跳过此模块，反之不满足则运行包涵的模块

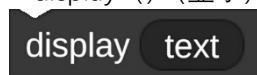
8: slse (否则)



此模块需要和 if <> then 或 else if <> then 连用，达到和以上程序相反的效果，其他性质一致，如以上模块条件达成则自动跳过 else 模块，反之亦然。

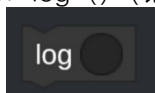
注:必须连用，且不能间隔

9: display () (显示)



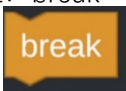
可以显示文字或者数字，圆框内可以直接填写文字，填入矢量模块会显示矢量 (x,y,z)，也可以填入计算公式如 (1) + (1) 等，如果公式有错会显示 NaN 意思是 No a number 没有这个数。

10: log () (记录)



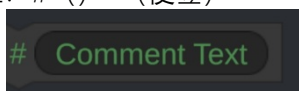
Log 和 display 大致相同，其不同点是 log () 可以将数据记录在 View Log 里，方便查看某一时刻的数值，当然 log () 将再把结果显示在屏幕上了

11: break (打破循环)



这将会跳出这个线程里包涵它的所有循环，直接执行最外层循环以下的模块

12: # () (便签)

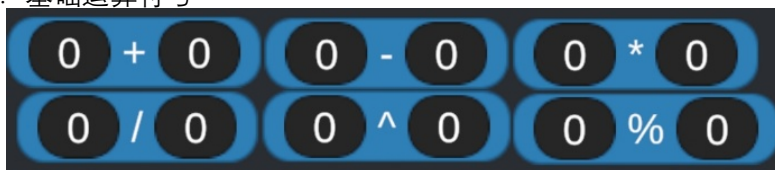


这个模块不会运行，你可以把它放在任何位置用于记录某些东西，你可以在里面自由填写，或是笔记，或是批注，你可以把它当成便签一样随意贴放
注:正版不支持中文，不要大惊小怪

二:operators (运算符)

这些模块包括了几乎所有的运算符，可以进行几乎所有的物理学运算，让你可以用 vizzy 完成玩家几乎不可能完成的任务。在此只简单翻译计算模块，这些模块包括了高中和大学的内容，需要些数学基础

1: 基础运算符号

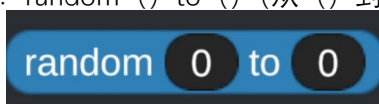


/是÷的意思， \wedge 是次方，如 $2 \wedge 3$ 就是 2^3

%: 取余 ($N \% D$) 就是将 N 分解为如下两部分之和:

$N = k * D + R$ 其中 R 就是余数

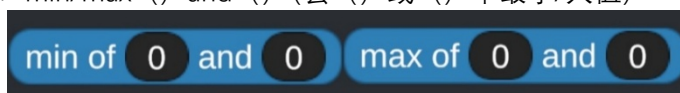
2: random () to () (从 () 到 () 随机取一个数)



随机的数几乎不可能是整数

注: 矢量不可用

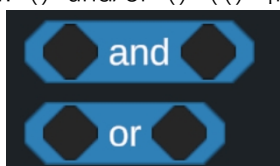
3: min/max () and () (去 () 或 () 中最小/大值)



取的是两个圆框内的其中一个值,

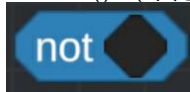
注: 矢量不可用

4: () and/or () (() 和/或者 ())



注意包涵关系，vizzy 会首先判断最内的模块

5: not () (不是 ())



当圆框内为否命题时整个 (not ()) 输出为真命题

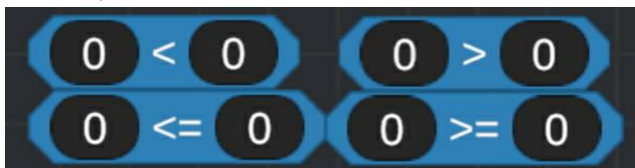
6: () = ()



等于“=”不仅可以用于数学，更多用于逻辑判断如: (part (0) Activited) = (ture) 的意思就 id 为 0 的部分 (零件) 的活动 (开关) 为是 (开) 则整个等于模块输出为真命题

由于 vizzy 刷新速度有限，如在游戏中 (AGL) = (1000) 海拔为 1000 的条件很难满载，因为上一帧可能是 999.9 下一帧就是 1000.5 了

7: 比大小



这就是简单的大于小于，如果满足大小条件则为真命题

8: []of () 函数运算



(1): abs 绝对值

(2): floor 取小于这个数的整数，也就是直接去掉小数点

(3): ceiling 取大于这个数的整数，直接去掉小数点再加一

(4): round 四舍五入去整数

(5): sqrt 算数平方根

(6): sin cos tan 三角函数，**输入为弧度制**输出为三角函数值的大小，如现有的是角度需要先转化为弧度制

(9): asin acos atan 反三角函数，和 (6,7,8) 相反，输入为三角函数值输出为**弧度制**

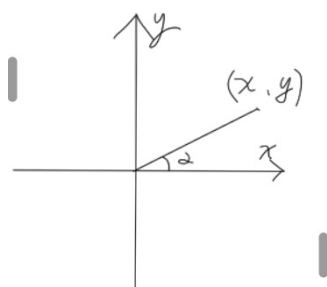
(10): ln 自然对数

(11): log 以 10 为底数的对数

(12): deg2rad 角度制转弧度制

(13): rad2deg 弧度制转角度制，两个不要记反了

9: atan2 of (y) and (X) 反三角函数



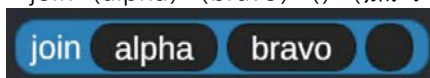
如图，模块输出的是 α 的弧度，结合上面的 rad2deg 便可得到相应的角度

10: if <> then () slse () (如果 <> 那么 () 否则 ())



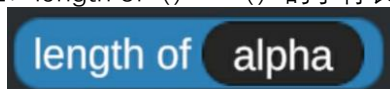
如果 <> 内的命题为真则输出为第一个 (), 如果 <> 为假命题则输出为后一个 (),
() 里可以填写 true 或者 false

11: join (alpha) (bravo) () (加入 (α) (β))



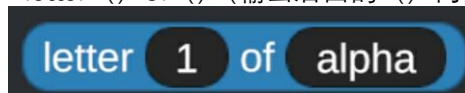
可以组成一个无空格的整体，如图输出的便是“alphabravo”由于无法输入单独的空格所以只能用英文负号隔开。可以用回车键提行，支持富文字

12: length of () () 的字符长度



数数，输出的是 () 的字符个数，如输入 (1,1,1) 输出的不是根号 3，而是 7 因为有 7 个字符

13: letter () of () (输出后面的 () 内的第 () 个字符)



如图，输出就是字母“a”

14: letter from () to () fo () (输出后面的 () 内的第 () 到第 () 个字符)



和上面同理，如图，输出为“lph”字符

15 () 包含 () (() 里是否包含 ())



如后面的框内包含前面的字符则输出为真命题，如图输出为真命题反之为假命题。

16: format () () (格式函数)

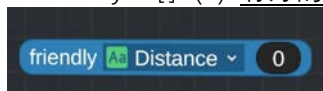


现已知两种函数 ({0:n1}) 和 ({0:e1})

1: {0:n1}意思是四舍五入到小数点后一位数

2: {0:ne}科学计数法保留一位有效数字

17: friendly [] (0) 有好的 在后面的数字后面添加所选择的物理符号。



翻译:以对用户友好的方式表示这个数，(给这个数字添加单位)

(1): Acceleration 加速度 m/s^2

(2): Angle Velocity 角速度

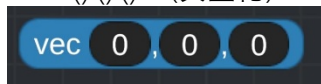
(3): density 大气密度 kg/m^3

(4): distance 离目标“target”的距离 m

(5): energy 能量 j

- (6): force 力 N
- (7): specific impulse 比冲 S
- (8): Mass 质量 kg
- (9): Power 电量 W
- (10): pressure 气压 Pa
- (11): Temperature 温度 k(凯尔文)
- (14): time 时间 s
- (13): Velocity 速度 m/s

18: vec() () () (矢量化)



将标量转化为矢量，图中三个空位分别对应 xyz，如图讲将出 (0,0,0)

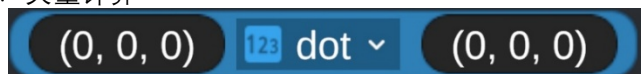
19: 矢量处理



对矢量进行处理

- (1): X; Y; Z 分别是选取矢量的第一个，第二个和第三个
- (2): length 模长
- (3): norm 取同方向上的单位长度，即模长等于 1

20: 矢量计算



矢量计算

- (1): angle (求角度) 以角度制输出浮点数;范围 0-180
- (2): clamp (钳制) 钳制前面括号内矢量的模长不超过后面的值，在前面函数内填入矢量 (x,y,z)，在后面的 (a) 内填入模长，若矢量模长大于后面的值输出的则是模长为 a 的和 (x,y,z) 同向的矢量，若小于则输出本身，后面的括号内只能填**纯数字**
- (3): cross 叉乘
- (4): dot 点乘
- (5): dist
- (6): min; mix 最大小模长
- (7): project 投影
- (8): scale

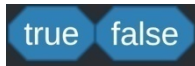
21: funk() funk 函数



函数暂无中文名。作用是可以拓展动作的输入范围（从 ag 1~10 和下面 input 模块的内容扩展到具体数学运算和你在零件 input 选项里见到的那些东西）。使用此函数注意两点：

- (1): 后面括号内只能填入**纯文本**，但函数本身会对文本内容进行计算并输出。比如：set slider1 to funk(0.1+0.2+0.3)，运行时程序会对文本 (0.1+0.2+0.3) 进行计算得到 0.6，然后把 slider1 设置为 0.6。
- (2): 此函数**只是拓展了动作的输入范围**，相当于一个数据接口。实际运行时还是以**零件的 input 选项的内容**为准。

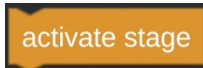
22: 真假命题



三: Craft Instructions 直译: 工艺说明

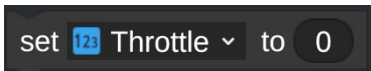
这些模块将会帮助你控制你的作品, 让你的作品更好的在太空中遨游, 他们能精确的控制每一个零件, 做到比手动更灵活和迅速的操作

1: activate stage 激活阶段



在游戏内依次激活当前以及往后的阶段, (V:0.9.802) 若非主要芯片某个线程出此模块, 这个线程会卡死不动

2: set [] to () 设置滑条 [] 到 ()



设置控制作品的滑条, 在此 1 表示 100%, -1 表示-100%

(1) roll 旋转

对应的是“Translation Mode Disabled 禁用平移模式”下右摇杆的左和右, 可以沿着作品向前的方向为轴旋转

(2) pitch 俯仰

对应的是“Translation Mode Disabled 禁用平移模式”下右摇杆的上和下, 可以沿着作品向前的方向俯仰

(3) Yaw 偏航

对应的是“Translation Mode Disabled 禁用平移模式”下左摇杆的对左和右, 可以沿着作品向前的方向水平偏转

(4) throttle 节流阀 (油门)

对应的是平面左端的黄色滑条或“Translation Mode Disabled 禁用平移模式”下左摇杆的上和下, 可以直接控制所开启的有油的油带正确的引擎的油门

(5) brake 刹车

对应的是“Translation Mode Disabled 禁用平移模式”下左摇杆的上和下, 但是需要在 throttle 为零的时候 brake 才会有读数, brake 可以控制轮子的刹车和有反推的涡轮引擎的反推

(6) slider 1 2 3 4 滑调 1 2 3 4

这是提给玩家自愿发挥的滑条, 可以在创造中调整零件的滑条

(7) translate Forward 平移向前

对应的是“Translation Mode Disabled 启用平移模式”下右摇杆的上和下, 可以沿着作品向前的方向前进和后退

(8) translate right 平移向前

对应的是“Translation Mode Disabled 启用平移模式”下左摇杆的左和右，可以沿着作品向左和向右

(9) translate up 水平向上

对应的是“Translation Mode Disabled 启用平移模式”下左摇杆的左和右，可以沿着作品向上和向下

(10) translate mode 平移模式

这不是滑条，在某些版本中如不打开平移模式 rcs 将不响应 vizzy 所定义的

Translate 滑条，0 或 1；false 或 true 表示禁用或启用

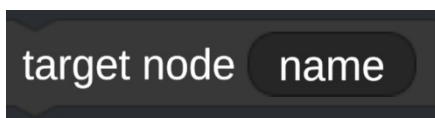
3: set craft [] to () 设置作品的俯仰或航向为 ()



设置作品的俯仰角 (1): pitch 或 (2): heading 航向，次模块讲自动为你打开“Locked Current Heading 锁定当前航向”并自动为你调整 pitch 和 yaw, 以使你的作品到达预定朝向，且不会调整 roll

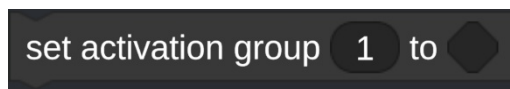
4: target node (name)

设置名为 (name) 的作品为 Target 目标



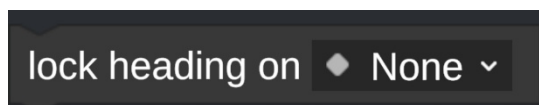
此模块将为你在屏幕上用白色框标出 Target 目标的位置和距离

5: Set activation group () to () 设置动作组 () 开或者关



此模块前面的括号里填入动作组的序号，游戏内默认是 1~10，但是可以在 xml 中设置更多，填写 1 或 0；false 或 true

6: lock heading on [] 讲飞船的航向设置为 []



此模块可以自动控制 pitch 和 yaw 以使你的作品的前方面对 [] 内所设置的目标。并不保证 roll 是否正常

(1): none 无

飞船将不会主动调整姿态，将会随受力情况沿重心自由转动，同时动量轮，指令仓，矢量喷口也不会工作

(2): prograde 向前

飞船将面对速度的正向

(3): retrograde 逆行的

飞船将面对速度的反方向

(4): Target 目标

飞船将面对所设置的目标，若未设置 Target 则锁定 none

(5): BurnNode 机动点

飞船将面对所设置的机动点，若未设置机动点则锁定 none

(6): current 当前的

锁定当前相对于母星的航向，并尽量保存不变

7: lock Heading on vector (x,y,x) 设置航向为矢量 (x,y,z)

lock heading on vector (0, 0, 1)

(0,0,1) 这里输入相对于作品质心的 pci 坐标，三轴分开没有意义，输入的坐标于与模长无关

8: set time mode to [] 设置时间模式为 []

set time mode to Normal

设置时间模式以使时间变快或变慢，全部在游戏中的作品将受此影响，在不同的时间模式下，作品的受力情况将不变

(1): paused 暂停的

在此暂停下所有 vizzy 也会停止

(2): SlowMotion 慢动作

(3): normal 正常的

(4): fastForward 快进的

(5-15): timeWarp1-10 时间扭曲

9: set camera [] to () 把用户的视角 [] 设置为 ()

set camera 123 Zoom to 100

控制玩家的视角

(1): x rotation x 轴旋转

(2): y rotation y 轴旋转

(3): tilt 倾斜

(4): zoom 变焦

设置用户观察作品的距离，单位为米

(5): camera mode 相机模式

切换玩家观察作品的模式，在 () 内填入相应观察模式的名字

(6): camera Index 摄像机索引

10: set part () [] to () 控制零件 () 使 [] 调整为 ()

set part 0 T/F Activated to 0

精确的控制某一个零件，在前面的括号里填入零件数字的 ID，不可以填入零件名称，他将无法识别名称。

(1): activated 激活 在后面的括号内填入 true 或 false，0 或 1 以表示开或

(2): name 名字 改变零件的名称，后面填入需要改变的名字。

(3): explode 爆炸 使填入的零件爆炸，后面的括号内填入爆炸等级，0-1 表示 0-100%，大于 1 也只有是 1 的效果，也可以是填 true 它和 1 等效

11: switch to craft () 转换到名为 () 的作品

switch to craft 0

玩家的视角和操作将转换到 () 作品上去, 换句话说, 此模块将使玩家操作名为 () 的作品

12: beep (1000) Hz at (1) volume for (2) seconds

beep 1000 Hz at 1 volume for 2 seconds

如图则, 发出 (1000) 赫兹声音大小为 (1) 的哔哔声持续 (2) 秒, 此模块将会以存在此模块并正在运行的零件为中心以大小为 (1) 的声音发相应的响声

13: frequency of (c#) octave (3) 频率为 (c#) 八度为 (3)

frequency of C# octave 3

这个模块可以被套在“beep () Hz at () volume for () second”中的 》Hz at () volume 《 中, 如图将发出标准的 C 大调升 mi 的音

四: Craft Information 作品信息

这些模块将输出模块所在的整体的响应信息, 可以使用这些信息进行计算从而得到有用的数据

1: altitude [] 海拔高度 []

altitude 123 AGL v

以米为单位输出对应的高度,

(1): AGL 相对于地面的高度

(2): ASL 相对于海平面的高度

2: orbit [] 轨道 []

orbit 123 Apoapsis v

这个模块将会根据飞船相对于母星的运动状况输出对应的有关于轨道的数值

(1): apoapsis 远地点

这是飞船相对于母星最远的距离, 也是线速度最慢的地点, 简称 ap 点

(2): periapsis 近地点

这是飞船相对于母星最近的距离, 也是线速度最快的点, 简称 pe 点, 但入大地图上无法显示出 pe 点, 此模块将输出“NAN”, 例如在火箭起飞的时候, 还没有形成环形轨道的情况下, 此模块将显示“NAN”

(3): time to apoapsis 到远地点的时间

飞船沿当前运动状态和受力情况, 到达远地点的时间, 通常可以简写为 ap 点

(4): time to periapsis 到达近地点的时间

飞船沿当前运动状态和受力情况, 到达远地点的时间, 通常可以简写为 pe 点

(5): eccentricity 离心率

显示以母星为焦点的椭圆形轨道的离心率, 输出值为 0-1

(6): inclination 斜率

(7): period 周期

输出以秒为单位的轨道周期

3: atmosphere [] 大气 []

输出飞行器所在的位置的大气信

atmosphere 123 Air Density ▾

(1) air density 空气密度

单位是 kg/m^3

(2): air pressure 气体压强

(3): speed of sound 音速

(4): temperature 温度

单位是开尔文 K

4: performance [] 性能 []

performance 123 Mass ▾

飞船有关性能的信息

(1): Engine thrust 最大引推力

当前正在工作的所有发动机的推力直接加起来，注意你的发动机可能有**夹角**，实际推力将不是输出值

(2): mass 质量

(3): dry mass 干重

飞船除去燃料后的质量，若修改了质量可能出现负数

(4): fuel mass 燃料质质量

不会随着 xml 中质量的修改而改变

(5): max Engine thrust 最大引擎推力

这是所有激活的引擎的推力直接加起来，不包含关闭的引擎推力，注意你的发动机可能**有夹角**，实际推力最大将不是输出值！涡轮引擎是当前高度和速度下的引擎推力，**无论引擎是否有燃料**只要启动都会参与计算

(6): TWR 推重比

飞行器推力和质量的比值，换句话最大加速度

(7): current Isp 当前比冲

(8): stage Delta-V 当前阶段的速度变化量

当前阶段的燃料和引擎所产生的速度变化量，不会计算大气对此的影响，会随节流阀的大小和大气情况产生变化。

(9): stage burn time 当前阶段燃烧时间

当前阶段所打开的引擎满载的燃烧时间，会随节流阀的大小和大气情况产生变化。

5: fuel [] 燃料 []

fuel 123 Stage ▾

以百分百的形式输出剩余燃料的值

(1): battery 电池

(2): stage 阶段的

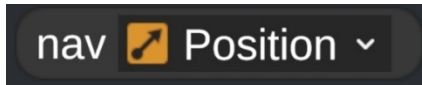
当前阶段剩余的化学燃料的百分比

(3): mono 单元燃料

(4): allstages 所有阶段

这是整个飞行器所剩的燃料百分比，空燃料箱分离后将回到 100%

6: nav [] 导航 []



此模块包含有关飞行器运动的信息，此模块输出为关于正在操作的飞船的质心关于所在行星的 PCI 坐标，xyz 分开几乎没有意义。

(1): position (坐标) 飞行器相对于母星的坐标，表示的是母星地心到飞行器质心的连线。

(2): Target position (目标坐标) 目标飞行就是相对于母星的坐标，表示的是母星中心到目标飞行器的连线。

(3): heading (航向) 输出的是 0~360 度，表示飞行器飞行的方向，0 度对应北面

(4): pitch (俯仰角) 飞行器和水平面的夹角，垂直于地面为+90 度，垂直于地面向下为-90 度，与海平面平行为 0 度，输出的值为角度的值。

(5): bank angle 倾斜角度

(6): angle of attack (攻角) 飞行器相对于空气速度的夹角。

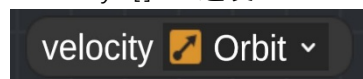
(7): side slip 侧滑

(8): north 北方

(9): east 西方

(10): craft roll/pitch/yaw axis (作品旋转/俯仰/偏航的轴的矢量) 以作品为原点的坐标，craft roll axis 就是垂直于 roll 轴的矢量坐标。这个矢量的方向跟指挥舱或者芯片的方向没有关系，若默认芯片为 rocket，在游戏开始时确定 roll 的方向就为垂直向上，此后随作品的运动而改变。

7: velocity [] 速度



飞行器有关速度的信息

(1): surface 表面的

这是相对于母星地面的速度 (PCI 矢量)

(2): orbit 轨道的

这是当前的轨道的线速度 (PCI 矢量)

(3): target 目标的

这是目标的地表速度，不是和目标的相对速度 (PCI 矢量)

(4): gravity 重力的

相对当前母星的重力加速度，不是母星的表面的重力加速度，此模块的输出是实时计算的 (PCI 矢量)

(5): acceleration 加速度

当前受力情况产生的加速度，包括重力加速度 (PCI 矢量)

(6): angular 角速度

飞行器自旋的角速度，不是轨道的角速度，xyz 分别对应 pitch yaw roll (PCI 矢量)

(7): lateral 水平的

平行于地面的速度分量 (纯数字)

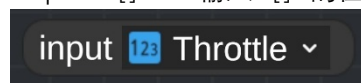
(8): Vertical 竖直的

垂直于地面的速度分量 (纯数字)

(9): mach number 马赫数

相对母星地面的速度的马赫数，声速会随空气密度变化而变化（纯数字）

8: input [] 输入 [] 的值



得到滑条的值，输出 1 ~ -1，与 set 同理

(1): roll （旋转）对应的是“Translation Mode Disabled 禁用平移模式”下右摇杆的左和右，可以沿着作品向前的方向为轴旋转

(2): pitch （俯仰）对应的是“Translation Mode Disabled 禁用平移模式”下右摇杆的上和下，可以沿着作品向前的方向俯仰

(3): Yaw （偏航）对应的是“Translation Mode Disabled 禁用平移模式”下左摇杆的对左和右，可以沿着作品向前的方向水平偏转

(4): throttle 节流阀（油门）对应的是平面左端的黄色滑条或“Translation Mode Disabled 禁用平移模式”下左摇杆的上和下，可以直接控制所开启的有油的油带正确的引擎的油门

(5): brake （刹车）对应的是“Translation Mode Disabled 禁用平移模式”下左摇杆的上和下，但是需要在 throttle 为零的时候 brake 才会有读数，brake 可以控制轮子的刹车和有反推的涡轮引擎的反推

(6): slider 1; slider 2 （滑条 1.2）这是提给玩家自愿发挥的滑条，可以在创造中调整零件的滑条

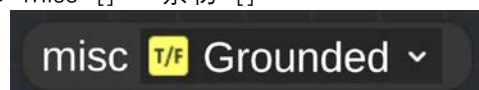
(7): translate Forward （平移向前）对应的是“Translation Mode Disabled 启用平移模式”下右摇杆的上和下，可以沿着作品向前的方向前进和后退

(8): translate right （平移向右）对应的是“Translation Mode Disabled 启用平移模式”下左摇杆的左和右，可以沿着作品向左和向右

(9): translate up （水平向上）对应的是“Translation Mode Disabled 启用平移模式”下左摇杆的左和右，可以沿着作品向上和向下

(10): translate mode （平移模式）这不是滑条，填入哈希值，表示禁用或启用 false 或 true 在某些版本中如不打开平移模式 rcs 将不响应 vizzy 所定义的

9: misc [] 杂物 []



判定飞行器的飞行状况

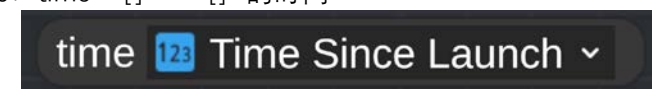
(1): stage （阶段）当前的阶段数，游戏开始时为 0

(2): nus stage 阶段总数

(3): grounded （触地的）输出为 false 或 true，触地则为真命题，输出为 true

(4): solar radiation 太阳辐射

10: time [] [] 的时间



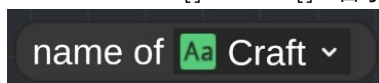
一秒为单位输出相应的时间

(1): frame delta time

(2): time since launch （发射后的时间）以秒为单位，输出距离第一次点火的时间

(3): total time （总时间）从开始游戏到现在的时间

10: name of [] [] 名字



以文字的格式输出 [] 的名字

(1): craft 作品

(2): planet 星球

飞行器所在引力范围内的星球

(3): target name 目标飞行器名称

(4): target planet 目标星球的名字

11: activation group (1) 动作组 ()

activation group 1

在后面的括号内填入动作组序号, 当动作组被激活此模块输出为真命题, 反之亦然, 在判定的模块中的 < > 内不用填写“activation group=true”, 因为当动作组被激活时此模块输出的就已经是“true”了。

12: convert () to lat/long/AGL 转化 () 到经/纬/高

convert (0, 0, 0) to lat/long/AGL

在 () 内填入坐标矢量“position”, 此模块将会以角度的形式输出所在引力范围内的星球的“lat/long”经纬, 和以米为单位输出高度, 海平面高度为 0, 最终输出的格式为 (精度, 纬度, 高度), 的一个带括号的矢量

13: convert () to position 转化 () 为 PCI 坐标

convert (0, 0, 0) to position

在 () 内填入经纬度和高度, 将输出相应的 PCI 坐标, 图像意义是星球球心到所填的经纬地度表示的位置的连线

14: get terrain [] at lat/long () 得到经纬度为 () 的地方的地势

get terrain 123 Height ~ at lat/long (0, 0, 0)

在 () 内填入经纬坐标, 输出的是作品所在引力范围内的行星的信息, 三为坐标将会自动省略最后一个。

(1): height (高度) 以米为单位输出海拔高度

(2): color (颜色) 以十六进制色的形式输出地表的颜色

15: planet () [] 得到行星 () 的 []

planet Droo 123 Mass ~

以正确的格式输出行星的信息

(1): Mass 质量, 单位 kg

(2): Radius 半径, 单位 m

(3): atmosphere height 大气高度, 单位 m

(4): solar position 太阳坐标

以太阳为母星的矢量坐标, 图像意义是一个连接太阳和行星的连线

(5): Child planets 卫星名字

以文字的格式输出行星的卫星的名

(6): Crafts 作品

以列表的形式输出引力范围内的作品的名字

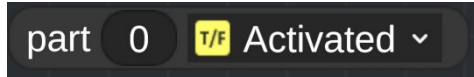
(6): crafts IDs 作品 ID

以列表的形式输出引力范围内的作品的 ID

(7): parent 母星

以文字的格式输出行星的母星的名字

16: part () [] 零件 () 的 []



在 () 内填入零件的 ID, 无法识别名称

(1): mass 质量

以 kg 为单位输出零件质量, 包括燃料

(2): activated 开关状态

根据零件是否被打开, 输出 false 或 true

(3): part type 零件类型

以文字的格式输出零件类型

(4): position 坐标

输出单个零件的 PCI 坐标位置

(5): temperature 温度

以开尔文为单位输出零件的温度

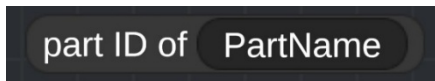
(6-7): min/max part ID ID 最小/大的同名零件

此模块的 () 内不能直接填写 ID 名称, 需要插入“part ID of ()”, 在“part ID of ()”内填入零件名称

(8): under water 在水下的

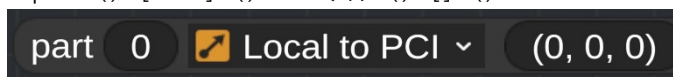
输出 0-1 以表示零件在水下的百分百

17: part ID of () 叫 () 的零件的 ID



() 内填入零件的名称, 将输出对应的 ID 号码

18: part () [to] () 零件 () [] ()

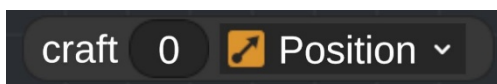


转化某零件或坐标的坐标系到零件或某坐标, local: 本地的, 以飞船作为参考系, XYZ 分别表示 yaw, pitch, roll 轴作为坐标轴, PCI: 在模块中指相对于母星的 PCI 坐标, 图像意义是一个从母星中心到飞行器零件的连线, 只是转化了参考系表示的矢量不变

(1): PCI to local

(2): local to PCI

19: craft () [] 作品 () 的 []



在 () 填入作品 ID, 输出作品信息

(1) altitude 海拔高度

(2) destroyed 是否被摧毁的

(3) grounded 触地的

- (4): mass 质量
- (5): name 名字
- (6): part count 零件数量
- (7): planet 行星

作品所在的引力范围内的行星

- (8): position 坐标 (矢量)
- (9): Velocity 速度 (矢量)

20: craft ID of () 名为 () 的作品 ID

在 () 内填入作品的名字, 输出的则为作品的 ID 号码

craft ID of **CraftName**

五: events 事件

有关游戏进行的模块

1: on start 在开始的时候

on start

万物的开头, 在游戏开始的时候便只激活一次此模块下面的第一个模块

2: on (part) collide with (other) at (Velocity) and (impulse)

当以零件 (part) 和 (other) 以 (velocity) 的速度和 (impulse) 的冲量撞击

on **part** collide with **other** at **velocity** and **impulse**

每一次撞击此模块都会激活, 并赋予里面的黄色参数新的数值, 里面的参数可以被拖动并调用, 用法和自定义变量相同。

3: on (part) exploded 当零件 (part) 爆炸的时候

on **part** exploded

每一次爆炸就会激活, 并刷新变量 (part) 为零件 ID

4: on (craftA) and (craftB) docked

当飞船 A (craftA) 和飞船 B (craft) 对接的时候

on craftA and craftB docked

5: on enter (planet) SOI 进入行星 (planet) 的引力范围内

on enter planet SOI

进入新的行星轨道便激活一次，刷新参数 (planet) 为行星名称

6: receive (message) with (data) 收到信息 (message) 的时候

当收到特定信息 (message) 时激活一次，刷新参数 (data) 为用户所设置的文本或数字

receive message with data

字

7: broadcast (message) with data (0) 在零件里广播 (message) 和 (0)
在整个零件里广播，只有此零件内的程序会做出回应

broadcast message with data 0

8: broadcast (message) with data (0) to craft
在整个飞船里广播 (message) 和 (0)

broadcast message with data 0 to craft

在整个飞船里广播，只有此飞船的零件内的程序会做出回应

9: broadcast (message) with data (0) to nearby crafts
在飞船周围广播 message 伴随和(0)

broadcast message with data 0 to nearby crafts

可以通过这个模块实现不同作品之间的通信

六: variables 变量

设置变量以使程序变得更简洁明了，简化重复的运算以优化程序，或做到更多的复杂运算。

建立变量，选择 variable 部分，点击左下角“create variable”在弹窗中输入变量的名称，或字母或数字只要自己认识，点击“create”完成创建，此时 variable 列表中会出现绿色的变量模块，这个模块可以被插入到 ()

修改变量，每一个绿色的变量模块左边有一个笔的标志，表示修改，点击会出现弹窗，“cancel”：返回 “rename”：重命名 “delete”删除

1: set variable () to (0) 设置变量 () 为 (0)

set variable to 0

在第一个 () 中填入变量的模块，变量模块将会变成一个输出后面 (0) 内的数字，或文字的模块，变量在游戏开始的时 by 候默认 0

2: change variable () by (0) 用 (0) 改变变量 ()

直接在变量 () 添加后面的字符，例如“change variable (X) by (0)”假设变量为 0 的

change variable by 0

话变量将会变成“00”

3: set variable () user input (message text...) 用户输入变量 ()，文本 ()

在前面的括号套入变量模块，后面的 (message text...) 填入文本，游戏中运行至此将会

set variable to user input Message Text...

出现一个弹窗以让用户输入变量。

七：lists 列表

我们可以把列表理解为**集合**，首先需要说明四个概念，即**列表 list**，**元素（项目）**，**索引（位置）**以及**长度 length**，我们举一个栗子，有一个班，班里有很多学生，那么我们手上的名单就是**列表**，名单上的人就是**元素**，他们的学号或者座位号就是**索引**，班上的总人数就是列表的**长度**

设置列表可以方便地对多个对象或者具有对应关系的对象进行处理，或者反过来将需要的数据按一定规则进行存储而不用用户手动记录。目前为止能明确的是列表都是单行。

列表里的元素和索引是一一对应的，列表的本质就是表格，

建立列表，首先选择 lists 部分，点击左下方“Create Lists variable”进行创建，在弹出的对话框内填入你想要的列表名称，字母数字都可以，只要自己能认识就行。点击“CREATE”完成创建，创建完成后，列表部分内最下方会出现紫色的列表模块，可以被插入到()。

修改列表名称与上方修改变量名称完全一致。

另外，列表非常友好的一点是，需要**填入元素**的地方都会写 item，需要**拖入列表**的地方全部是空的，而填写列表**索引**的地方全部写上数字 1。，**列表的位置（索引处）从 1 开始，到列表长度结束的整数**。请在继续往下看时牢记这几个规则。

注：下列凡是括号内写“**列表**”的都是填入**列表名称**，写“**1**”的都是填入**数字**，写“**元素**”的都是填入**元素**。

1: add (item) to () 为列表 () 添加元素 (item)

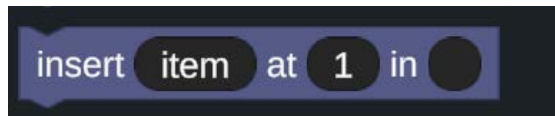
add item to

第一个空格里填入你要添加的元素还有，元素可以是数字，矢量，字符串或者某个**变量**或者一个确定的字符，第二个空格里填入列表的名称。例如：

有列表 [1,2,3,4,5] 通过 >add (ABC) to (列表名称)< 列表将变成[1,2,3,4,5,ABC]

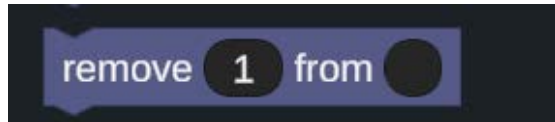
Add 一次只能添加一个项目，如在 (item) 处填写 >A,B< 之类的无论如何也只会把填入的作为一个项目处理

2: insert (item) at (1) in () 在(列表)中的索引处(1)插入(元素)



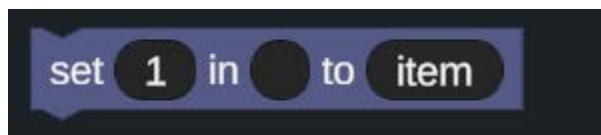
比如列表[1,2,3,4,5]通过 >insert (item) at (3) in (列表名称) < 列表就变成[1,2,item,3,此外和 add 一样只能处理单个项目

3: remove (1) from () 将索引为 (1) 的元素从(列表)移除



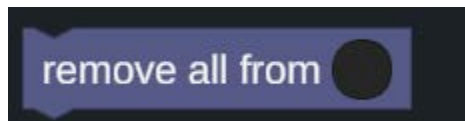
在有 1 的 () 内填索引，后面留空的 () 内拖入列表名称。比如列表[1,2,3,4,5]的索引处为 3 的元素就是 3，将其删除后，列表变成[1,2,4,5]。

4: set (1) in () to (item) 将 (列表) 索引处 (1) 设置为 (元素)



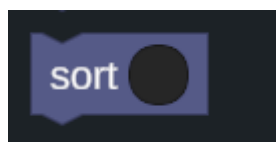
就是将列表中列表位置 (1) 的原来的元素替换为 to 后面的东西。

5: remove all from () 将 (列表) 中的所有元素清除。



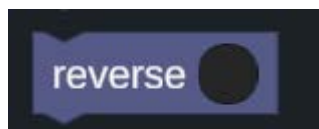
字面意思，把指定的一个列表内的元素清空。

6: sort () 将 (列表) 中的所有元素按照升序序列重新排序。



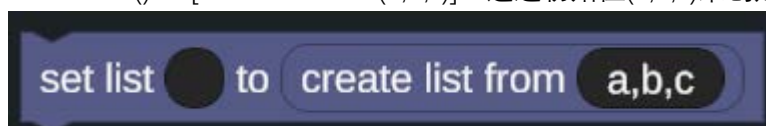
字面意思，把一个列表中的元素从小到大或者 a 到 z 的顺序排列。

7: reverse () 前后颠倒 (列表) 中的所有项目。



字面意思，把指定的列表中的所有元素前后颠倒。注意，这个地方只是把元素前后颠倒，没有按从大到小排序。例如列表[a]={a,c,d,e,b,f}, reverse [a]={f,b,e,d,c,a}, 并没有从大到小排列。

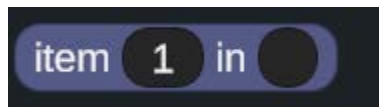
8: set list () to [create list from(a,b,c)] 通过初始值(a,b,c)来创建 (列表)。



这里是通过直接指定列表内元素是什么来创建列表。一般用在已知列表内元素的情况下。注意与 add (item) to ()的区别。add (item) to ()可以不知道元素具体值就直接添加入列表，

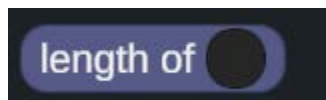
而 `set list () to [create list from(a,b,c)]`则必须已经明确知道列表元素为何。

9: `item (1) in ()` (列表) 中 (1) 的元素



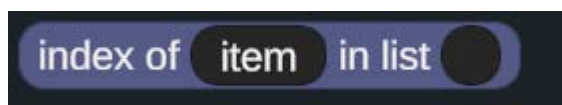
这是一个获取列表指定位置的元素的函数，与上面运算符部分一样，不能作为单独的一个步骤，而是作为参数使用。

10: `length of ()` (列表) 的长度



获取列表元素个数，也就是列表长度的函数

11: `index of (item) in list ()` (元素) 在 (列表) 中的位置

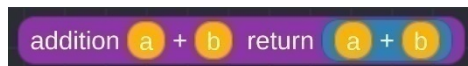


获取 (元素) 在 (列表) 中的索引值，也就是某一个元素在列表中的位置。。如果在列表中找到该项目，返回 0。比如[1,2,3,4]中元素 3 的索引值是 3, 也就是第三个。元素 0 在列表中不存在，则返回值 0。

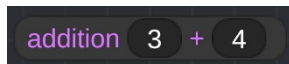
八：Custom Expressions 自定义表达式

自定义表达式作用是将程序内复杂的自定义运算或重复调用的运算等独立出程序，使程序更加整洁，更改参数更加方便。创建方式参考下面的自定义指令。

点击左下角 **create Custom Expressions** 会出现弹窗，并附有“**enter a name of Custom Expressions**”意为，输入自定义表达式的名称，这里可以输入任何自己喜欢的字符，如图输入的是“addition”，接下来会出现一个新的弹窗，底部会出现“**cancel**”取消 “**add text**”添加文本 “**add parameter**”添加变量 “**create**”创建，达到如图效果，便是先点击的“**add parameter**”添加变量，并输入字符“a”确认后，点击“**add text**”添加文本，并输入字符“+”再点击“**add parameter**”添加变量，输入字符“b”最后点击“**create**”创建，系统会自动为你插入一个“**return**”并生成一个圆形空位 () ,可填入运算符和变量。



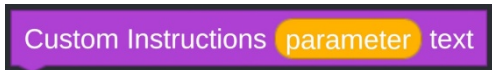
这样就完成了表达式的创建。调用时，变量从本栏中拖出，“0”的地方分别填入你所需要的数字，对应的变量将会被赋予对应的值，如图“a”将被赋值“3”，“b”将被赋值“4”



程序运行时会直接将数字进行运算，如图模块将输出“7”，因为 3+4=“7”

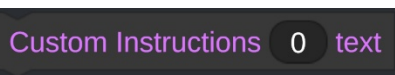
九：Custom Instructions 自定义指令

自定义指令可用于自定义需要执行的指令顺序，但它们只会运行一次，它们类似于由多个模块组成一个新的模块。创建之后，这个自定义模块将放置在您的 custom instructions 的目录中，这样您就可以定义这个自定义指令应该执行的确切指令。另外，将在左侧的“自定义指令”面板中创建一个新的指令模块，以便您可以用于在程序中调用/执行此自定义指令



创建自定义指令，点击左下角 create Custom Instructions 会出现弹窗，在窗口中输入自定义指令的名称，点击 create 便会出现一个新的弹窗，此时弹窗底部会出现“cancel”取消 “add text”添加文本 “add parameter”添加变量 “create”创建。

若要达到如图效果，点击“add parameter”输入自定义参数的名称，在自定义指令的模块中会出现一个黄色的椭圆形自定义参数模块，再点击“add text”输入文本，最后点击“create”完成创建，便可达到如图效果。

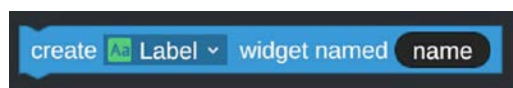


调用，完成创建后在 custom instructions 的目录中出现与之相对应的模块，你可以拖动并调用，在程序中会自动为你添加一个如（1）图一样的模块，拖动模块并在下面添加其他模块以合成新的模块。如（2）图的模块可以被放在程序当中来表示自定义模块，其效果和（1）图下面接的模块等效，其中于黄色的参数模块相同的位置对应的括号可以赋予该参数值，参数可以在（1）图的模块中被拖出并调用在自定义指令中。

十：multi-function display 多功能显示器

在“多功能显示器”这个零件里创造部件并显示，用 vizzy 调整大小；角度；坐标；透明度等等，来显示自己想要的图形或文字内容，多功能显示器简称“mfd”，在这里你可以显示任何东西，甚至 3D 图案，MDF 默认的小部件数量上限是 99 个，像素结构的上限是 265 个像素，其原因是防止程序中出现过于巨大的数，无限的生成模块导致设备卡死或宕机，可以在 xml 中修改。

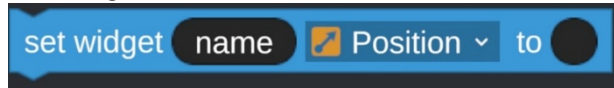
1: create [] widget named () 创造小部件[]名字叫 ()



- (1): ellipse 圆
- (2): label 标签
- (3): line 线
- (4): radial gauge 环形仪表盘
- (5): rectangle 矩形
- (6): texture 结构
- (7): vanball 导航球
- (8): map 地图

这是显示器的核心之一，**往后任何我们要定义的小部件都需要先创造它**然后在设置它

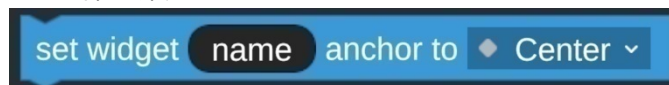
2: set widget () [] to () 把名字为 () 的小部件的 [] 设置为 ()



在前面创造了小部件以后，需要设置设置小部件的一些特性，当然如果不设置这些特性他们将会是默认值，这不是必要的东西

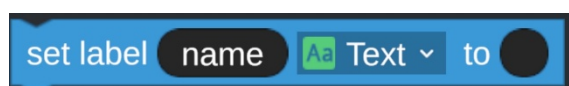
- (1): anchored position 绑定坐标
- (2): anchor mix
- (3): anchor min
- (4): color (颜色) 颜色是十六进制色例如#FFFFFF 将会是黑色，如果直接填写颜色将必须加上“#”，如套上 (hex color ()) 在 (hex color ()) 内可以不加
- (5): opacity (不透明度) 在后面的 () 内填写 0-1 以表示不透明度，0 表示完全透明，1 则为完全不透明
- (6): parent 起源
- (7): pivot 旋转中心
- (8): position (坐标) 平面坐标，X,Y，可以不带括号，0,0 是显示器的中心
- (9): rotation 旋转
- (10): scale (规模) 1 表示 100%大小
- (11): size (大小) 1 表示 100%大小
- (12): visible (看得见的) True 或 false 1 或 0 真命题表示看得见，反之亦然

3: set widget () anchor to [] 把名为 () 的小部件绑定在 []
没什么卵用



- (1): left 左面
- (2): center 中间
- (3): right 右边
- (4): toplift 左上
- (5): topcenter 中上
- (6): topright 右上
- (7): Bottomlift 左下
- (8): bottomcenter 中下
- (9): bottomright 右下

4: set label () [] to () 设置标签为 [] 填入 ()



设置所创造的“label”部件以显示一些内容

(1): text 文字

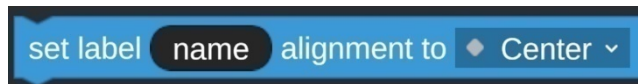
(2): font size 文字大小

填入 1-0, 1 表示 100%大小

(3): auto size 自动大小

True 或 false 启用或禁用自动大小, 自动更改文本的大小, 以填充标签小部件的全部区域

5: set label () alignment to [] 把名字为 () 的标签对齐于 []



(1): left 左面

(2): center 中间

(3): right 右边

(4): toplift 左上

(5): topcenter 中上

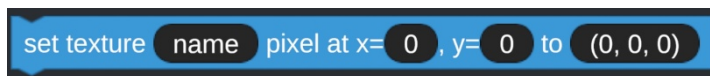
(6): topright 右上

(7): Bottomlift 左下

(8): bottomcenter 中下

(9): bottomright 右下

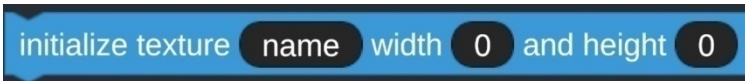
6: set texture () pixel x= () y= () to ()



设置名为 () 的结构位于 x= () y= () 的像素为 ()

设置所创造的结构像素, 在后面的 (0,0,0) 中填入十六进制颜色, 如果直接填写颜色将必须加上“#”, 如套上 (hex color ()) 在 (hex color ()) 内可以不加

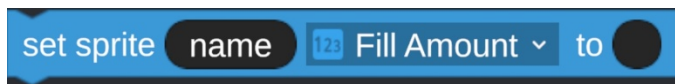
7: initialize texture () width () and height ()



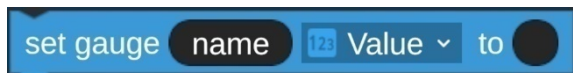
初始化名为 () 的结构使其宽为 () 高为 ()

在设置所创造的结构之前需要先初始化,

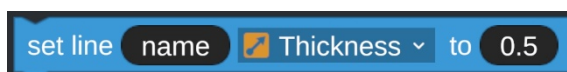
8: set sprite () [] to ()



9: set gauge () [] to ()



10: set line () [] to () 设置名字为 () 的线的 [] 为 ()



设置之前创造的小部件线的一些特性

(1): thickness 厚度 (宽度)

0-1, 1 表示 100%

(2): length 长度

11: set line () from () to () 设置名字为 () 的线从坐标 () 到坐标 ()

set line name from 0,0 to 1,1

把之前创建的线设置为一个从坐标 (0,0) 到 (1,1) 线

12: set navball () [] to 设置名为 () 的导航球 [] 部位的颜色为 ()

设置所创造的导航球, [top color] 上面的颜色; [bottom color] 下面的颜色为 (), ()

set navball name Top Color to

中填入十六进制颜色, 如果直接填写颜色将必须加上"#", 如套上 (hex color ()) 在 (hex color ()) 内可以不加

13: set map () [] to

设置所创造的地图的一些特性, 不必要, 程序中没有这个模块将会是默认值

set map name North Up to true

(1): North Up 以北为上

启用或禁用地图锁定向北, 启用后地图将不会转动, 表示用户的图标将会转动, 禁用反之亦然

(2) zoom 缩放

设置地图的缩放级别。这只能是介于 1 和 5。1 表示 100%

(3) manual mode 手动模式

手动模式下是固定地图的, 不能在游戏里面设置或改变地图

(4) planet 行星

在手动模式下选择地图所显示的行星

(5) coordinates 坐标

在手动模式下选择地图所显示的坐标, 在 () 中的填入经纬度

(6) heading 朝向

在手动模式下条件选择地图的朝向

14: bring widget () in front of () 把名为 () 的部件带到 (Target) 前面

bring widget name in front of target

15: send widget () behind () 把名为 (name) 的部件送到 (Target) 附近

send widget name behind target

16: set widget (name) to broadcast (message) on [Click] with data (data)

将名为 (name) 的小部件设置为在 [Click] 点击时广播 (message) 并赋值 data 为 (data)

set widget name to broadcast message on Click with data data

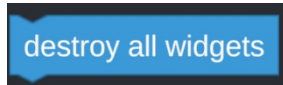
在名为 (name) 的小部件被点击时, 广播 (message) 和赋予黄色的 (data) 模块值, 此模块需要和上面的“五-6”配合使用

17: destroy widget named (name) 清除名字叫 (name) 的小部件

destroy widget named name

可以防止卡顿，及时把不必要的部件删除，若只为隐藏也可以把不透明度调为 0，或者设置隐藏为真命题

18: destroy all widgets 清除所有部件



防止卡顿

19: get widget (name) [position] 得到名为 () 的部件的 []
跟上面的 set 同理一个是，设置一个是得到



- (1): anchored position 绑定坐标
- (2): anchor mix
- (3): anchor min
- (4): color 颜色

颜色是十六进制色例如#FFFFFF 将会是黑色，如果直接填写颜色将必须加上“#”，如套上 (hex color ()) 在 (hex color ()) 内可以不加

- (5): exists 存在

如果具有指定名称的小部件存在，则输出为真命题

- (6): opacity 不透明度

在后面的 () 内填写 0-1 以表示不透明度，0 表示完全透明，1 则为完全不透明

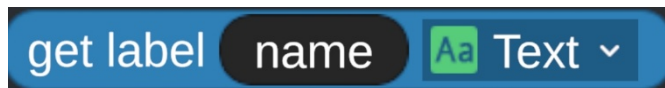
- (7): parent 起源
- (8): pivot 旋转中心
- (9): position 坐标

平面坐标，X,Y，可以不带括号，0,0 是显示器的中心

- (10): rotation 旋转
- (11): scale 规模
- 1 表示 100%大小
- (12): size 大小
- 1 表示 100%大小

- (13): visible 看得见的

20: get label (name) [] 得到名字为 () 的标签的 []
得到



- (1): text 文字

输出文字信息

- (2): font size 文字大小
- 输出 1-0, 1 表示 100%大小
- (3): auto size 自动大小
- 输出为真命题，或假命题
- (4): alignment 对齐
- 输出为对齐的地方的名称

21: get sprite () [] 得到名为 () 的圆的 []

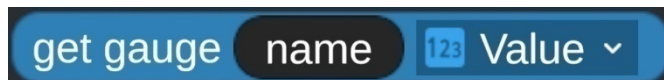


得到圆的信息

- (1): Fill method 填充方式
- (2): icon
- (3): fill amount 自动填充

根据是否禁用自动填充输出对应的真假命题

22: get gauge () [] 得到名为 () 的仪表盘的 []
得到所创建的仪表盘的一些特性



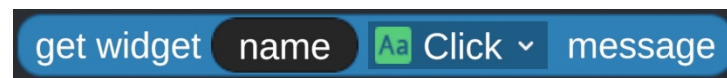
- (1): background color 背景颜色
- (2): Color 颜色
- (3): text 文字
- (4): text color 文字颜色
- (5): value 值

就是仪表盘指的数值 0-1, 1 表示 100%

23: convert (0,0) [] for ()



24: get widget () [] message 自己理解



25: hex color () 颜色



十六进制颜色, 可以不带#号

叁 程序实例

一：用 pid 让火箭悬停

pid 是一种常见的控制方式，有适应性强，响应快能通过当前状况实时调整等特点，适用范围非常广泛，但并非万能。这里以让火箭悬停为例来介绍 pid，程序如图



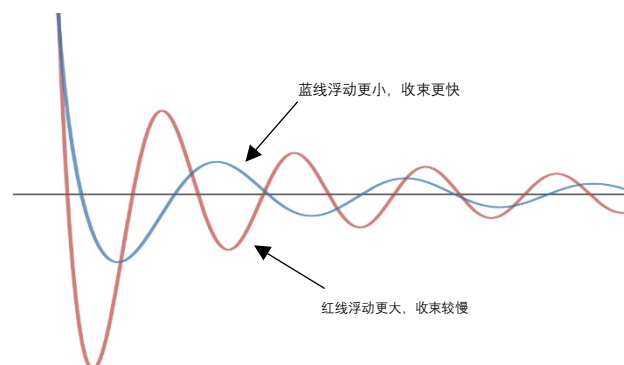
正如图所示，pid 分为三个部分，输出是由三个变量共同决定的，分别是 p i 和 d，英文的全称是 proportional-integral-derivative(放洋屁)，意为偏差的比例、积分、微分，虽然这看起来很高深，其实我们可以更通俗的理解这三个变量。

p(proportional)，这是误差量，如图我们定义的变量 target 是目标高度为 10 米，AGL 表示距地面高度，用目标减去当前的距地高度便得到了误差量。

I(integral)，直意为积分，意思是通过 p 和 d 的输出累加得到一个相对稳定的数值，而积分的意义是稳定当前的状态，在当我们知道如何去稳定当前状态的时候就不用大费周章的计算积分，如例所示，我们就深知只要平衡了火箭的重力便能维持运动状态，所以我们对 i 的定义就是火箭的重力。

d(derivative)，意为微分，顾名思义就是偏差改变的速率，若偏差改变的速率过快，换句话说就是一直加速前往目标，一定会冲过目标，冲过的量甚至会超过原来的误差，所以我们要根据误差改变的速率实时限制输出，在本例中我们要到达预计高度，误差改变的速度就是火箭竖直的速度，并在前面乘上负一，使其速度越快负得越大，对整体的抑制作用越强，便可有效避免了上述情况。

细心的你一定发现了在本例中 p 和 d 的后面乘了质量，其实这是系数，本例只是举例而选质量作为系数，在 pid 中系数尤为重要，是否合适直接决定了收束的速度，系数不仅可以是一个实数，也可以是一个表达式，这等待你自己去探索。



这是 pid 收束曲线，可以看见虽然加了积分限制输出，但是还是多多少少会冲过标准线，一些，其次显而易见蓝色线的收束速度高于红线，这便是系数不同的影响，这也提现了 pid

的局限性，越大的误差收束需要的时间成倍增长，所以 pid 只适合相对较小误差的微调。若是试图只用基础的 pid 着陆，那么火箭将重重的摔在地上！

总结来讲 p 是误差量，i 是一个想方设法去稳定当前状态的量，d 是限制输出的量

一：简单的锁定程序


利用矢量计算编写一个简单的锁定程序, 让你的摄像机或者灯或者其他东西朝向某个方向，本节包含了基本的矢量应用，叉乘的应用，以及一种图像思维



1:利用矢量绘图

游戏中的矢量都是有所对应的，都是可以被画出来的，请先阅读前 while 下面的前三行并试图绘制其图像（模块的翻译前面都有，涉及矢量的模块都标注了图像意义）

肆 其他有关内容

- 1: **矢量** vizzy 用  表示, 可以同时表示大小和方向的量, 在本游戏里以 (0,0,0) 的方式表示

PCI 坐标系: planet central inertial 天体中心惯性系, 在游戏里是一个坐标轴指向绝对坐标系, 关于某个天体 (你的作品也可以是天体) 的中心为零点, xyz 轴方向保持绝对不变与姿态无关, 如图 (31.1), 且相互垂直, 游戏中所有 pci 坐标对应的轴都平行且与相对的物体无关如图 (31.2) 及原点可变; 坐标轴指向不变, 所以每一个 pci 坐标都可以平移并直接运算, 注意:三个轴拆开来独立分析几乎没有价值

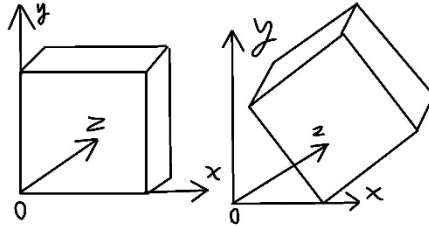


图 (31.1)

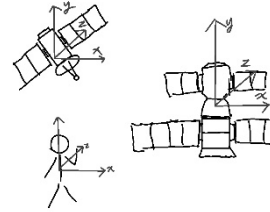





图 (31.2)

Local 坐标系 顾名思义本地坐标系 以分别以 pitch yaw roll 为坐标轴的 x y z 轴, 三者的对应关系不定, 在 vizzy 模块的详解里有标注

- 2: **浮点数** vizzy 中用  英文名 float, 含字节数为 4, 32bit, 数值范围为 $-3.4E38 \sim 3.4E38$ 的数, 小数点后有很多位, 直接 display 可能难以看清, 游戏里任何数据都是以浮点数储存的
- 3: **角度** vizzy 中用  表示, 这是 0-180 的浮点数
- 4: **弧度** vizzy 中用  表示, 弧度就是我们常说的几兀 (兀 $\approx 3.1415\cdots$), 1 兀=180 度
- 5: **Dv:** dv 是 Δv 的简称, 表示速度的变化量, 例如从静止变得以 10m/s 的速度位移, 则此过程中的 dv 就是 10m/s。
- 5: 列表 纯数字 文字 在 vizzy 中这是三个不同的概念, 不可混用, 处理文字的模块不能处理列表, 处理
- 7: <http://digitalnativestudios.com/textmeshpro/docs/rich-text/>

伍 Vizzy 的 XML

游戏内自带的编译器虽然编程方便但 bug 巨多可读性极差，在程序数量更多的情况下掉帧严重，一个核有难七核观战，面对这种情况我们就不得不直接在 xml 里面编写程序。

一：程序的储存和信息

```
<FlightProgram maxInstructionsPerFrame="250"
powerConsumptionPerInstruction="0.01" broadcastPowerConsumptionPerByte="0.1">
  <Program name="New Program">
    <Variables>
      <Variable name="Variable" number="0" />
      .....
    </Variables>
    # 程序主要内容
  </Program>
</FlightProgram>
```

如图，所有程序被放在了 FlightProgram 块里面，其中 variable 被放在 program 的最开头里面储存的是程序变量，后面的 number=0 是初始化值，但不能改。

二：事件类

1: 事件触发语句

```
<Instructions> #当游戏开始的时候
  <Event event="FlightStart" id="0" style="flight-start" pos="0,0" />
</Instructions>
<Instructions> #当飞行器发生碰撞的时候
  <Event event="PartCollision" id="0" style="part-collision" pos="0,0" />
</Instructions>
<Instructions> #当飞行器其有一个零件爆炸的时候
  <Event event="PartExplode" id="0" style="part-explode" pos="0,0"/>
</Instructions>
<Instructions> #当两个飞行器成功对接的时候
  <Event event="Docked" id="0" style="craft-docked" pos="0,0" />
</Instructions>
<Instructions>
  <Event event="ReceiveMessage" id="0" style="receive-msg" pos="0,0">
    <Constant canReplace="false" text="message" />
  </Event>
</Instructions>
```

6: 正如上面所写的，每一个 event 类型的代码都被包涵在一个 instructions 中，这相当于

java 中的一个类或者方法，在 vizzy 中每一条程序都由一个 event 引导所以 event 类型通常放在 `instructions` 下面的第一个，后面的 pos 是坐标，是 position 的简写，只有 event 和独立的模块会带有坐标，其他的代码可能会也有坐标，是因为程序抽风删掉即可不影响。

二：变量和计算

1：在读写变量之前需要先创建变量，用于储存变量的位置位于 program 下的第一个

```
<Program name="New Program">
  <Variables>
    <Variable name="Variable" number="0" /> #定义一个名为 Variable 的变量
    <Variable name="Variable1" number="0"/> #定义一个名为 Variable1 的变量
  </Variables>
  .....
```

2：给变量赋值改变和让用户输入

```
<SetVariable id="1" style="set-variable"> #将变量设置为数字 0
  <Variable list="false" local="false" variableName="VariableName" />
  <Constant number="0" />
</SetVariable>
<ChangeVariable id="2" style="change-variable"> #在变量后面添加一个字符 0
  <Variable list="false" local="false" variableName="VariableName" />
  <Constant number="0" />
</ChangeVariable>
<UserInput id="3" style="user-input"> #让用户输入变量数值并显示文字 message text...
  <Variable list="false" local="false" variableName="VariableName" />
  <Constant text="Message Text..." />
</UserInput>
```

3：计算 π 的近似值

$$\left(\frac{1}{1}\right)^2 + \left(\frac{1}{2}\right)^2 + \left(\frac{1}{3}\right)^2 + \left(\frac{1}{4}\right)^2 + \left(\frac{1}{5}\right)^2 + \left(\frac{1}{6}\right)^2 + \dots = \frac{\pi^2}{6}$$

```

<FlightProgram                                powerConsumptionPerInstruction="0.01"
broadcastPowerConsumptionPerByte="0.1">
  <Program name="New Program">
    <Variables> #创建变量
      <Variable name="π" number="0" />
      <Variable name="num1" number="0" />
      <Variable name="num2" number="0" />
      <Variable name="num3" number="0" />
    </Variables>
    <Instructions>
      <Event event="FlightStart" id="0" style="flight-start" pos="0,0" />
      <While id="1" style="while">
        <Constant style="true" bool="true" />
        <Instructions> #典中典的 while true 循环
          <SetVariable id="2" style="set-variable"> #num1 每个循环都+1
            <Variable list="false" local="false" variableName="num1" />
            <BinaryOp op="+" style="op-add">
              <Variable list="false" local="false" variableName="num1" />
              <Constant text="1" />
            </BinaryOp>
          </SetVariable>
          <SetVariable id="3" style="set-variable"> #(1/num1) ^ 2
            <Variable list="false" local="false" variableName="num2" />
            <BinaryOp op="^" style="op-exp">
              <BinaryOp op="/" style="op-div">
                <Constant text="1" />
                <Variable list="false" local="false" variableName="num1" />
              </BinaryOp>
              <Constant text="2" />
            </BinaryOp>
          </SetVariable>
          <SetVariable id="4" style="set-variable"> #把所有 num2 都加起来得 num3
            <Variable list="false" local="false" variableName="num3" />
            <BinaryOp op="+" style="op-add">
              <Variable list="false" local="false" variableName="num3" />
              <Variable list="false" local="false" variableName="num2" />
            </BinaryOp>
          </SetVariable>
          <SetVariable id="5" style="set-variable"> #num3 乘 6 开根号便是π
            <Variable list="false" local="false" variableName="π" />
            <MathFunction function="sqrt" style="op-math">
              <BinaryOp op="*" style="op-mul">
                <Variable list="false" local="false" variableName="num3" />
                <Constant text="6" />
              </BinaryOp>
            </MathFunction>
          </SetVariable>
        </While>
      </Instructions>
    </Program>
  </FlightProgram>

```

三：循

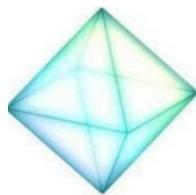
```
<Repeat id="1" style="repeat"> #重复 10 次
  <Constant number="10" />
</Repeat>
<While id="2" style="while"> #当 bool=true 的时候循环
  <Constant style="true" bool="true" />
</While>
<For var="i" id="3" style="for"> #for 循环 from () to () by ()
  <Constant number="1" /> #第一个数
  <Constant number="10" />
  <Constant number="1" /> #第三个数
</For>
<If id="4" style="if"> #如果循环
  <Constant bool="false" />
</If>
<Elseif id="5" style="else-if"> #else if 循环
  <Constant bool="false" />
</Elseif>
<Elseif id="6" style="else">
  <Constant bool="true" /> #else 循环
</Elseif>
```

1: 在每一个循环内部必须加 instruction 再写入循环内的程序主体 例如 for 循环 display。
Hellow world

```
<For var="i" id="1" style="for">
  <Constant number="1" />
  <Constant number="10" />
  <Constant number="1" />
  <Instructions>
    <DisplayMessage id="2" style="display">
      <Constant text="hellow world" />
      <Constant number="7" />
    </DisplayMessage>
  </Instructions>
</For>
```

尾页

主编：所罗门老狗 技术支持：普朗克的钟 カチューシャ



联系方式: [QQ:1853609050](https://t.me/1853609050) 邮箱: 1853609050@qq.com 或 tridentLaogou@gmail.com
若认真看过还有一脸懵逼的可以直接@作者，如果我在会出来回答并补充，当然也欢迎指正
错误或加入我一同编写等

群推荐，这些是一些讨论 sr2 的氛围非常好的群，有很多大佬可以互相帮助，当然作者也在
里面，这些都不是作者的群，甚至连管理都不是，作者不会帮某人宣传他的群：

530286238 简单火箭 2

979670763 simpleRockets2 简单火箭 2

490980588 简单火箭 2 贴吧群

转载须知：

- 1：禁止删除修改此页或加入自己的名字
- 2：禁止在本资料中擅自添加额外的宣传任何人或事物的文本或图片
- 3：禁止改变作者名称和联系方式
- 4：禁止用于除游戏外的其他用途
- 5：作者能力有限，若有明显错误可以适当删改 **除封面和尾页除外**

外部引用：

<https://wnp78.github.io/Sr2Xml/>

<https://baike.sogou.com/m/v675036.htm>

所罗门老狗 2020-07-24

修定日期：2022-7-15

V3.7.15