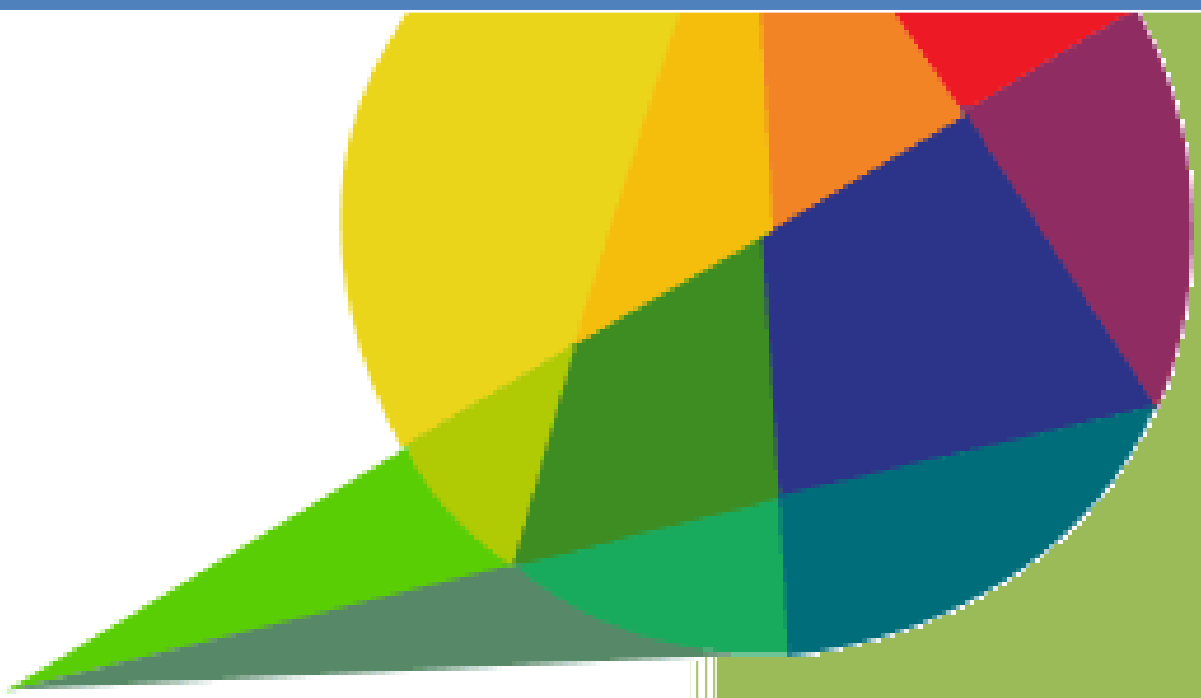
2018

迭代进阶（累次迭代）
---网络画板赛第 83 期分享

（江苏省苏州市吴中区木渎实验中学　　金晓亮）



作者：

成都景中教育软件有限公司

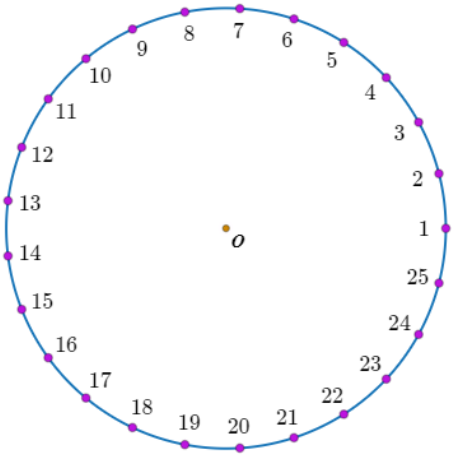
2018/7/9

网络画板的迭代功能是我所接触的所有画板（网板、几何画板、图霸和 ggb）中功能最强大的，因为它对迭代的初始条件限制很少，只要两个对象之间存在父子关系，都能进行迭代，而不需要要求原像必须是自由对象或者半自由对象。没有了这个限制，网板的迭代真的是相当容易。当然纯数据的迭代还不能实现。

本次分享主要是把我对迭代的理解讲解一下，我对迭代理解得也不是太深，可能有些地方也理解得不对，如果有错误之处可能在所难免，也欢迎大家多提意见，希望通过这样的交流使大家共同提高，共同进步。

第一部分　网络画板迭代原理

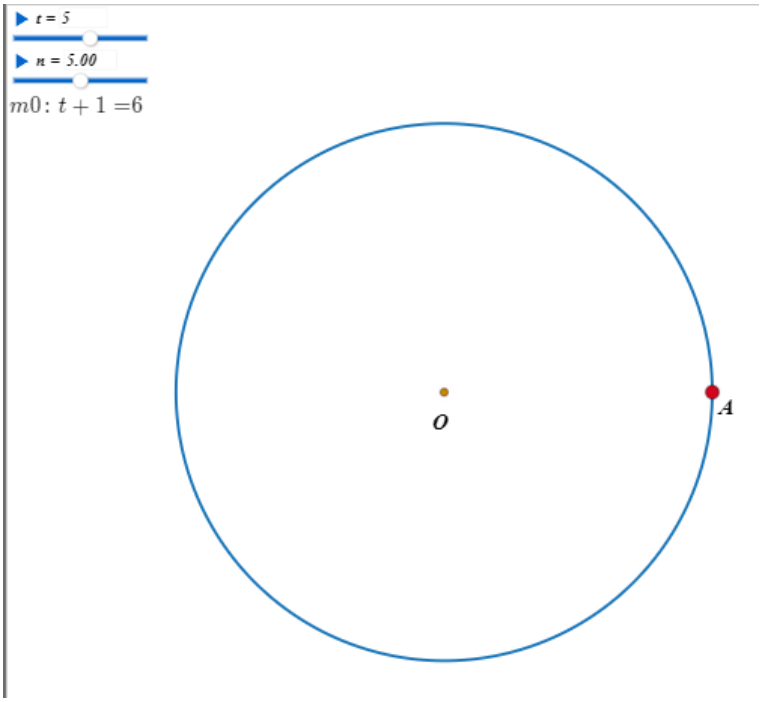
首先介绍一个简单的案例，现在想实现如图的效果，该如何迭代？



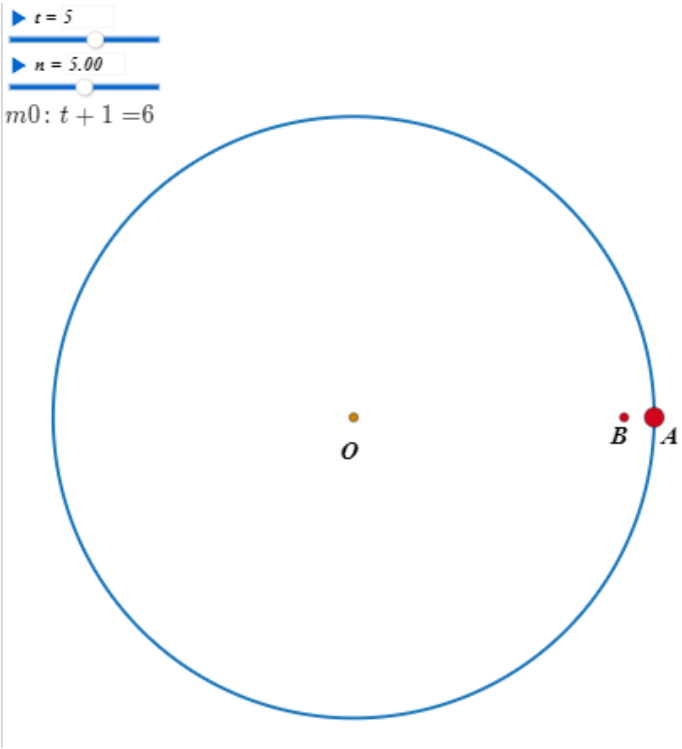
要实现这个效果，方法非常多，现在介绍如下：

首先共同的步骤如下：

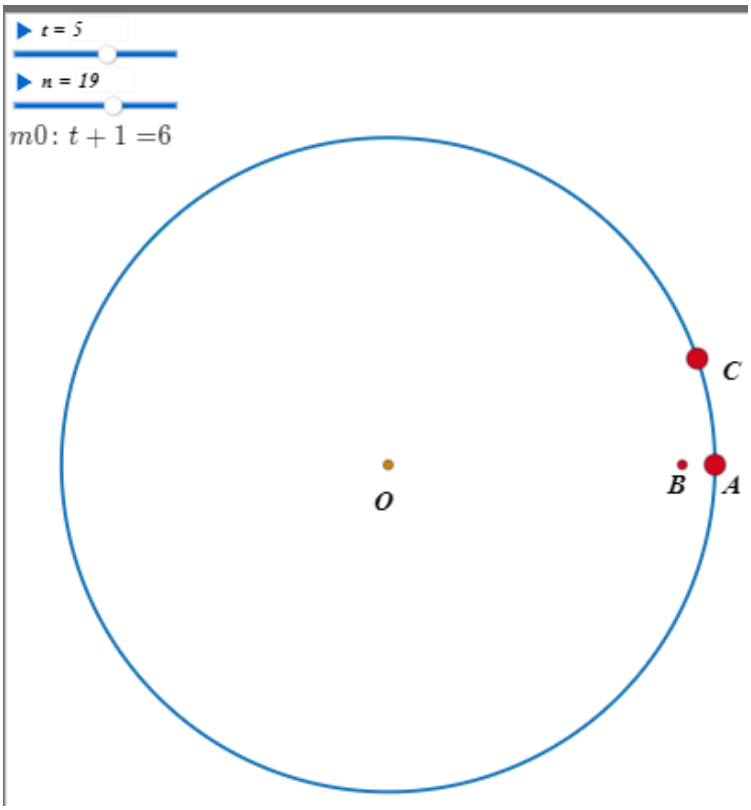
画一个圆 O，创建变量 t 和 n，计算 t+1



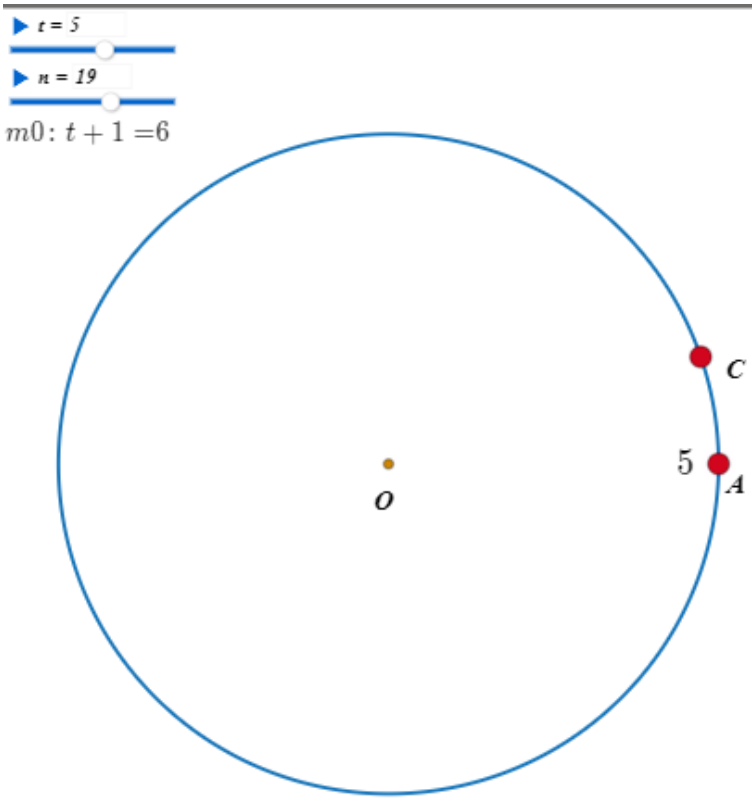
方法一：1.以点 O 为缩放中心，缩放比为 0.9 缩放点 A 得到点 B



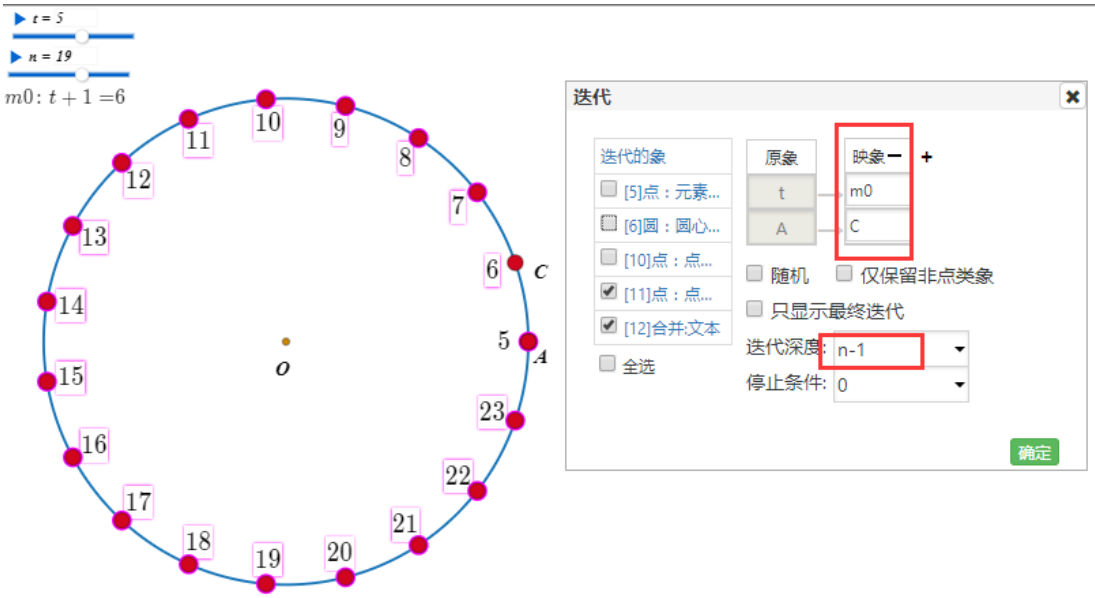
2.再以点 O 为旋转中心，旋转角度为 2pi/n，把 A 点逆时针方向旋转到点 C，



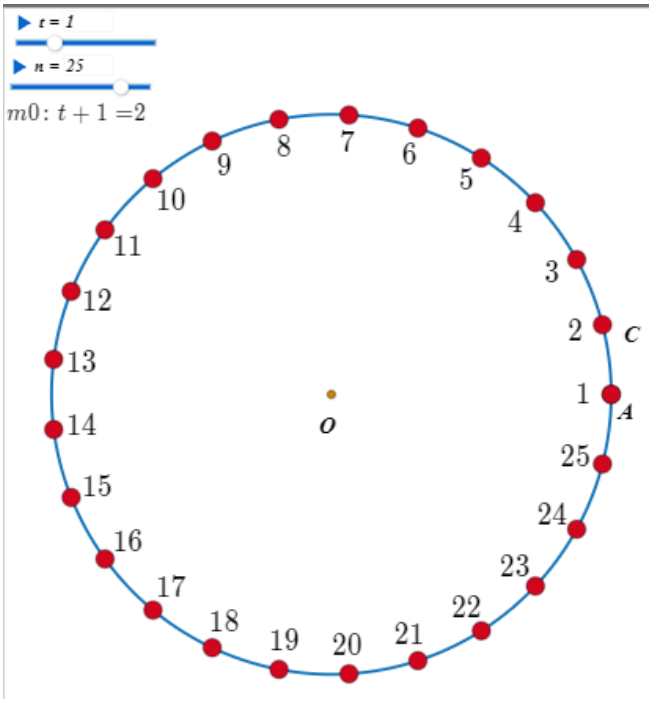
3.把变量 t 附着到点 B，并隐藏点 B，



4.选中点 A 和变量 t，进行迭代，如图



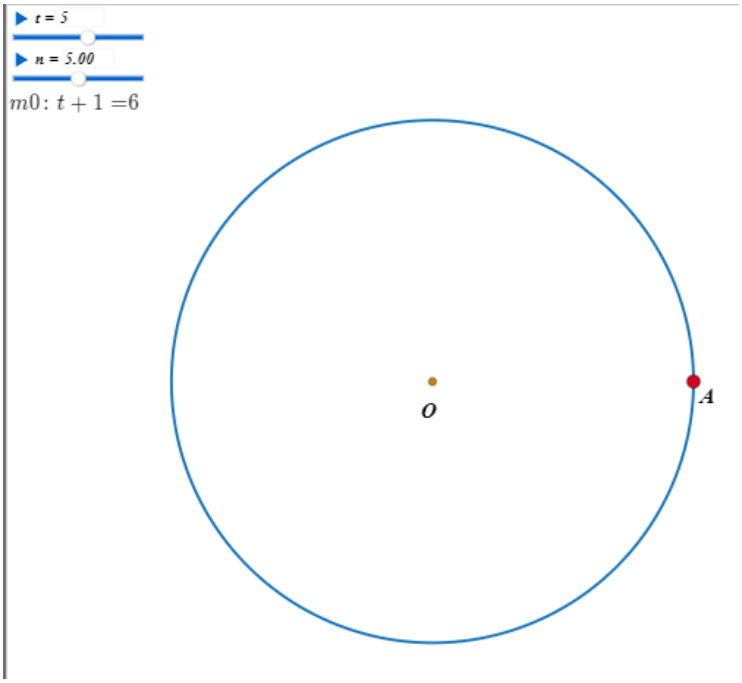
5.最后把变量 t 拖动到数值 1，变量 n 拖动到 25 即可。



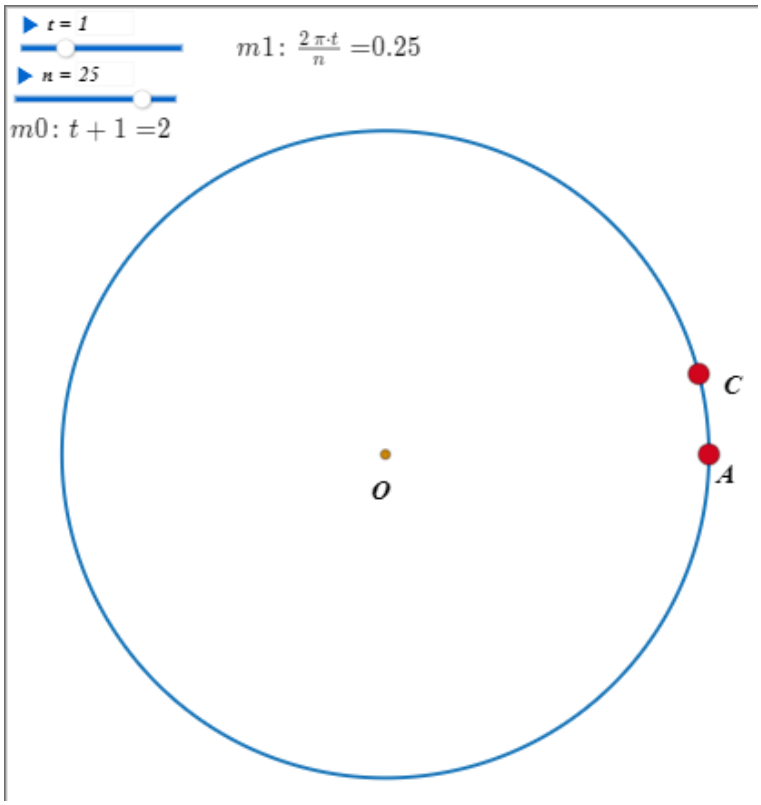
总结：这个方法的特点是原像既有点对象（点 A），又有数据变量（变量 t），而且全部图形包括两部分，一部分是原像（只有一开始的点 A 和一开始合并的标签 1，点 C 可以隐藏，不会影响图形的显示），另一部分是迭代像（就是标签从 2 开始一直到标签 25，以及包括对应的圆上的各点）。可见，总共生成 25 个标签，只需要迭代 24 次，还有一次就是一开始的原像对应的标签。

有时候，如果我为了统一起见，我不想借用原像来显示，而想整个显示的结果就是一个迭代的整体，那该如何实现呢？

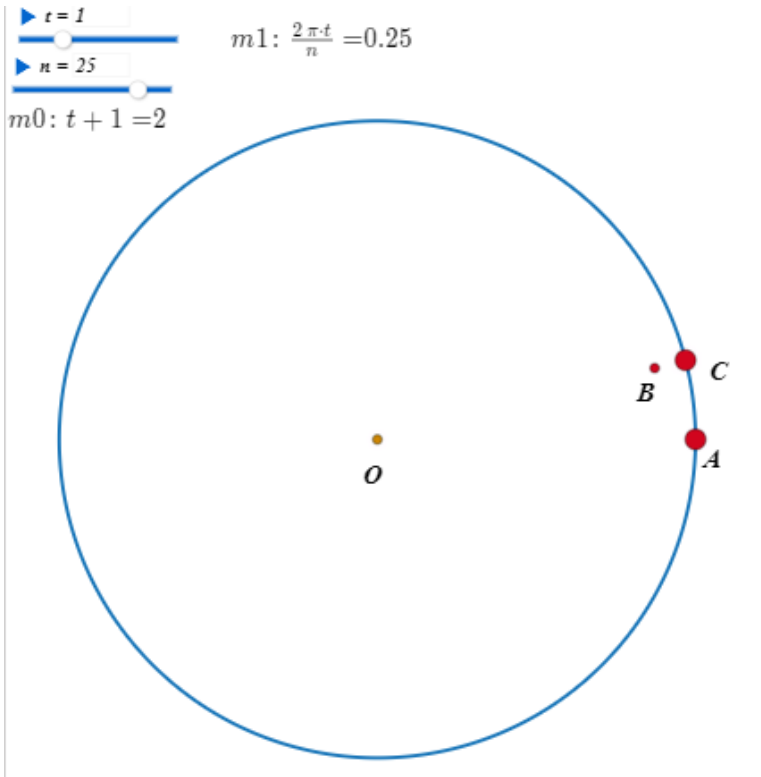
方法二：1.回到一开始的状态



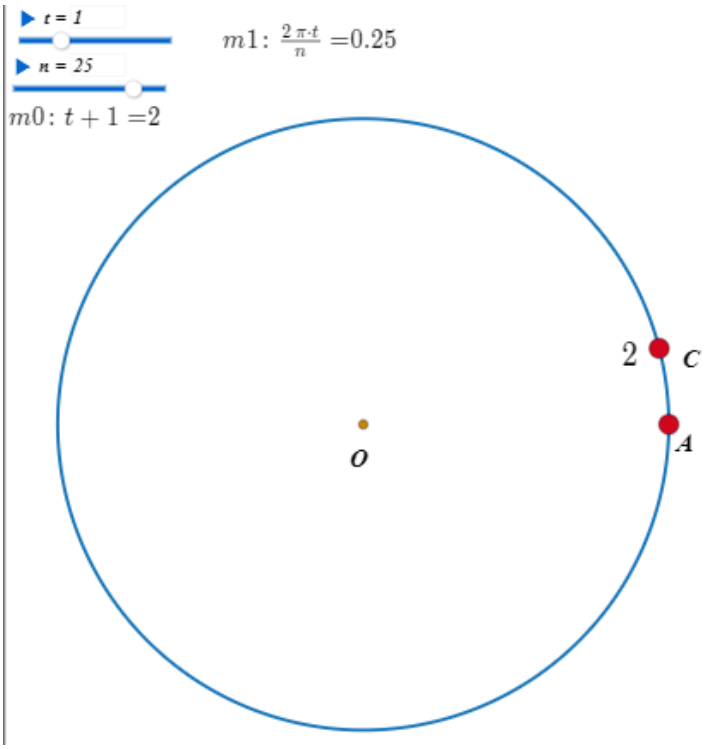
2.计算 2pi*t/n 的值，以 O 为旋转中心，计算值为旋转角度把点 A 逆时针旋转得到点 C，



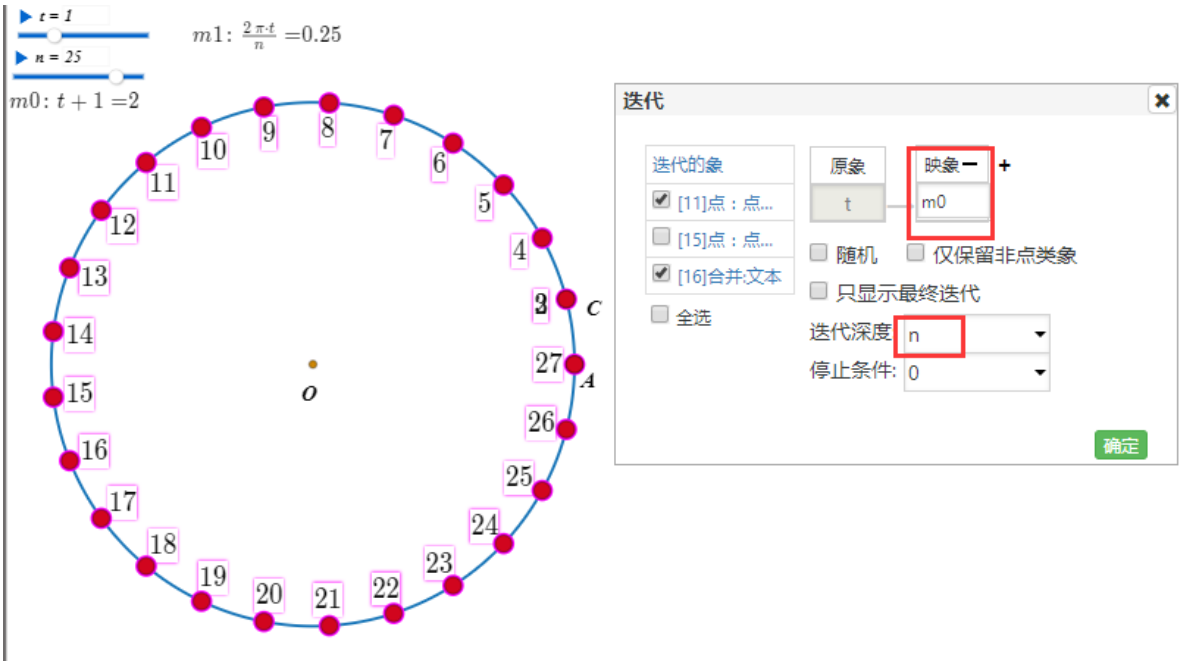
3.以点 O 为缩放中心，把点 C 缩放 0.9 得到点 B，



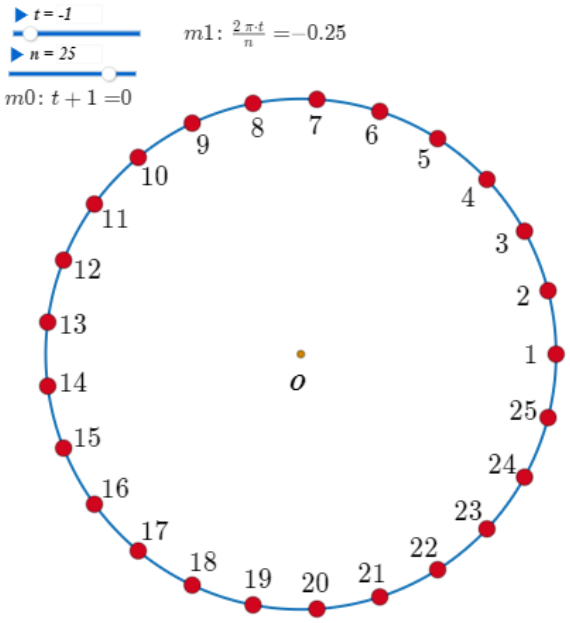
4.把计算值 m0 附着到点 B，并隐藏点 B，



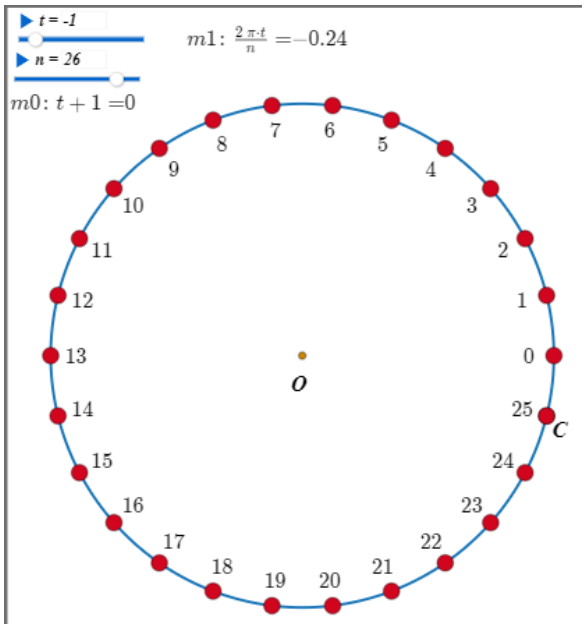
5.选中变量 t 进行迭代，如图



6.隐藏点 A，点 C 以及合并的文本，并把变量 t 拖动到-1

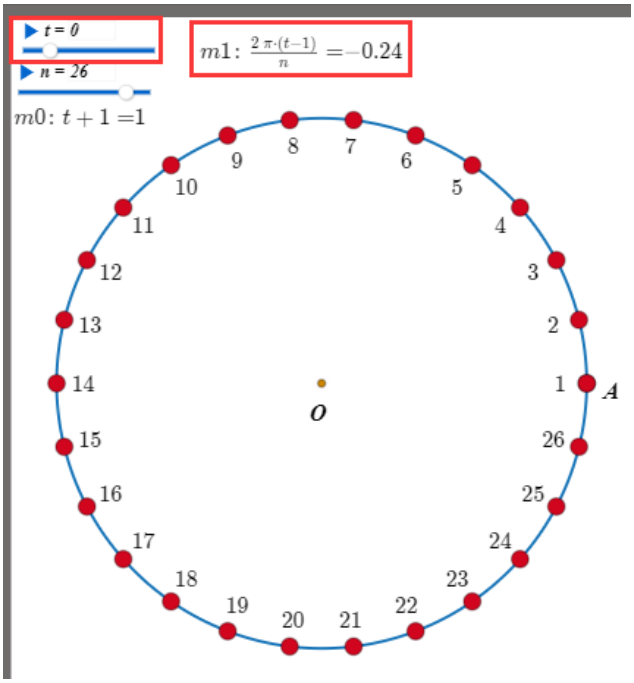


在上面的操作过程中的第 4 步，问什么要合并 m0 到点 B，为什么不能合并 m0 的原像 t 的值到点 B？我们不妨测试一下，结果就是这样：



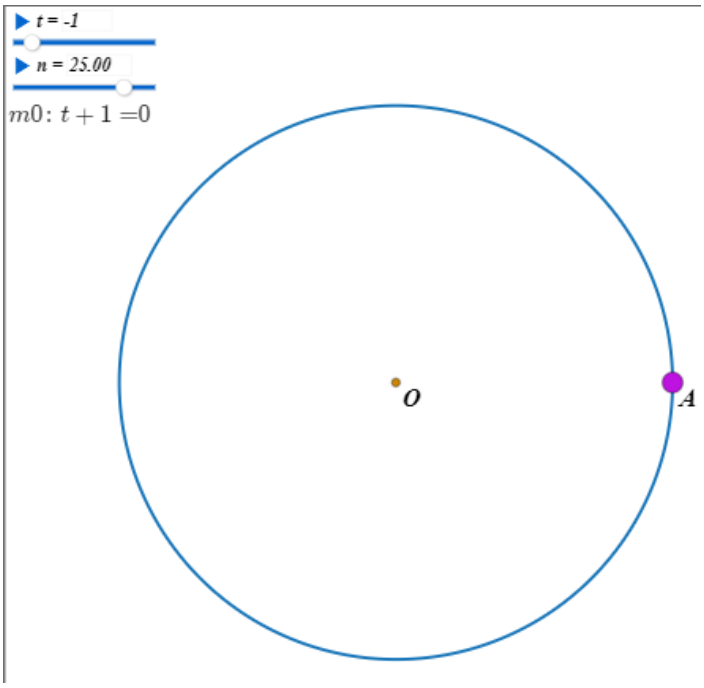
这两者的区别和联系是：首先迭代的初值值 t 都等于-1，下图中迭代 m 次的结果是上图迭代 m-1 次的结果。为什么会这样呢？因为上图是对映像 m0 进行迭代，它生成的第一个迭代像就是 m0+1，而下图是对原像 t 进行迭代，它生成的第一个迭代像就是 m0，它生成的第二个迭代像才是上面的 m0+1。**可见，当初始值不变的情况下，迭代原像产生的第 m 次结果就是迭代映像产生的第 m-1 次结果。**

那么如果我就是想把变量 t 的值合并到点 B，而也想产生想要的结果，有没有办法呢？答案是有的，我们可以修改 m1 的值变成：2pi*(t-1)/n ，并且让变量 t=0，如下图所示：

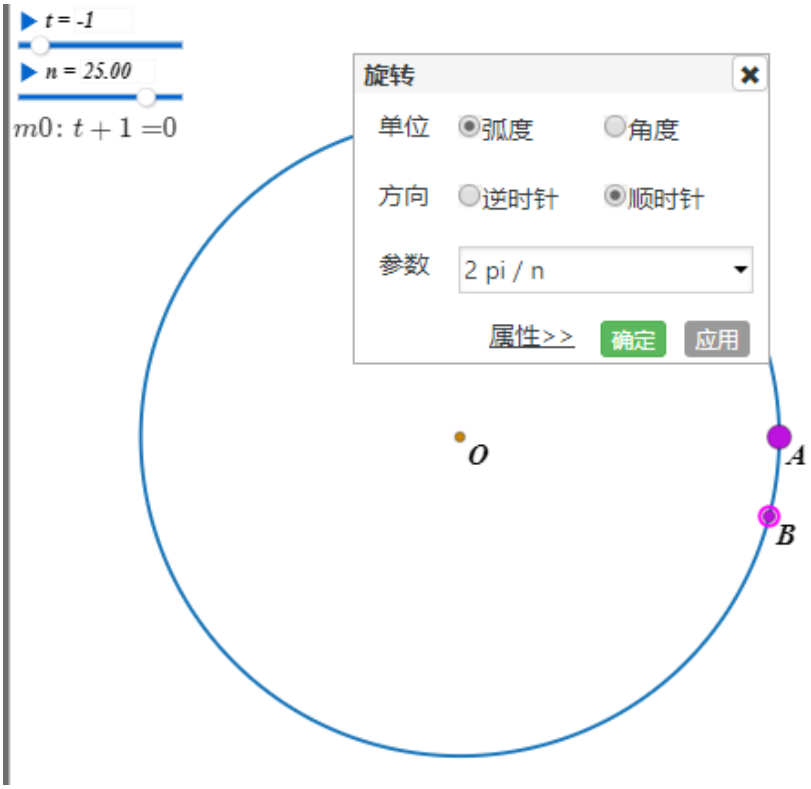


总结：这个方法的特点是，所有的图像都是迭代像，原像已经隐藏掉了，这样如果用变量 n 来控制迭代深度时，处理起来就可以实现统一，而不需要当 n=1 时再单独处理原像的显隐。这个方法的另一个特点是迭代时的原像只有变量 t，而没有点对象。

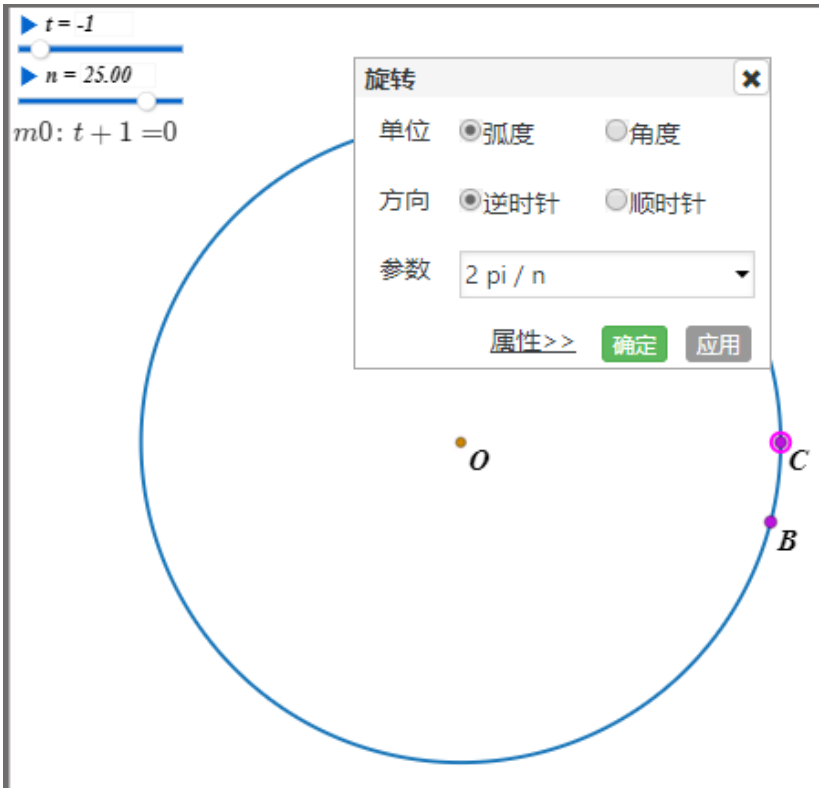
方法三：我们甚至还可以这么做，1.回到初始状态：



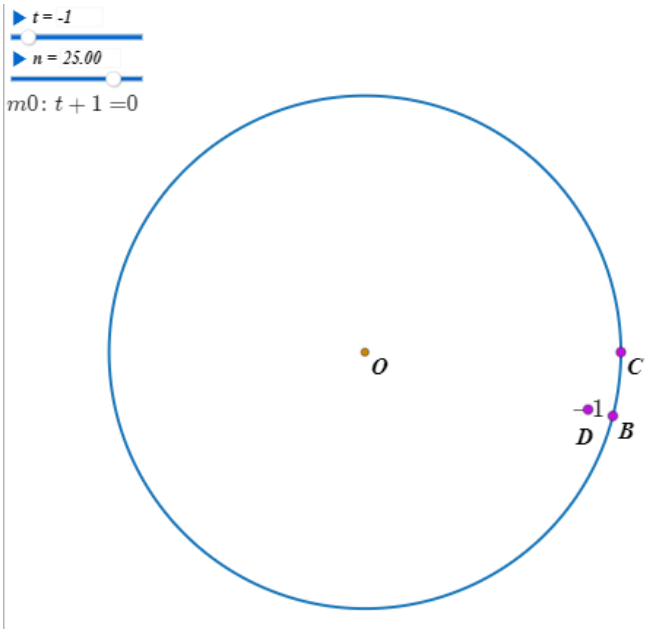
2.以点 O 为旋转中心，旋转角度为 2pi/n，把点 A 顺时针旋转得到点 B



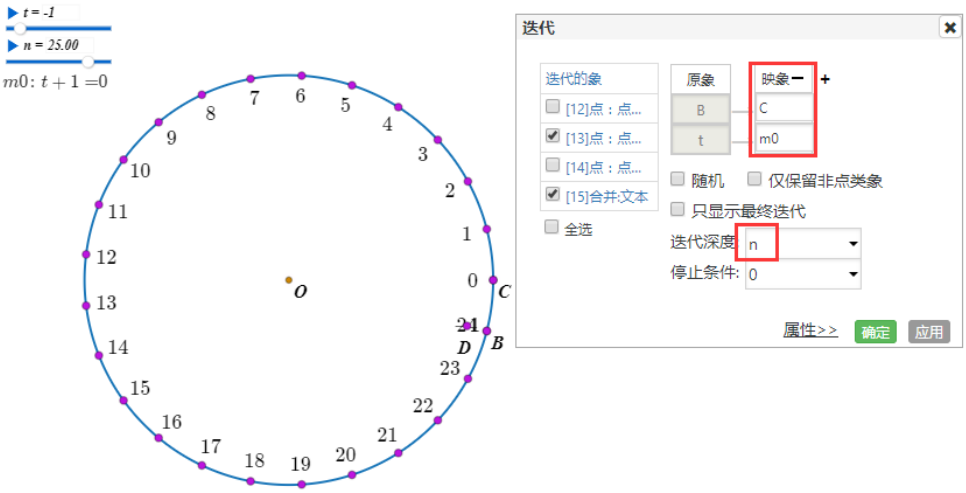
3.隐藏点 A，以点 O 为旋转中心，旋转角度为 2pi/n，把点 B 逆时针旋转得到点 C



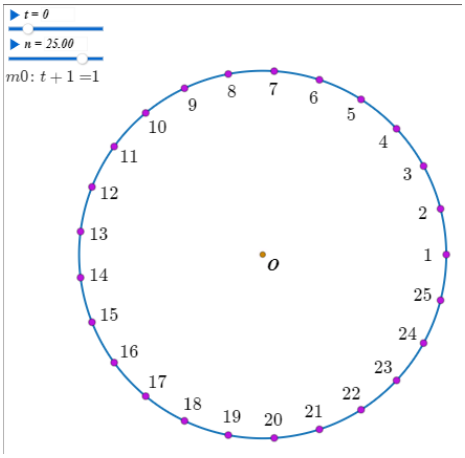
4.以点 O 为中心，缩放比为 0.9 缩放点 B 到点 D，并合并 t 的值到点 D



5.选中变量 t，点 B 进行迭代



6.把变量 t 拖动到 O，并隐藏 B，C 两点和合并的标签即可。

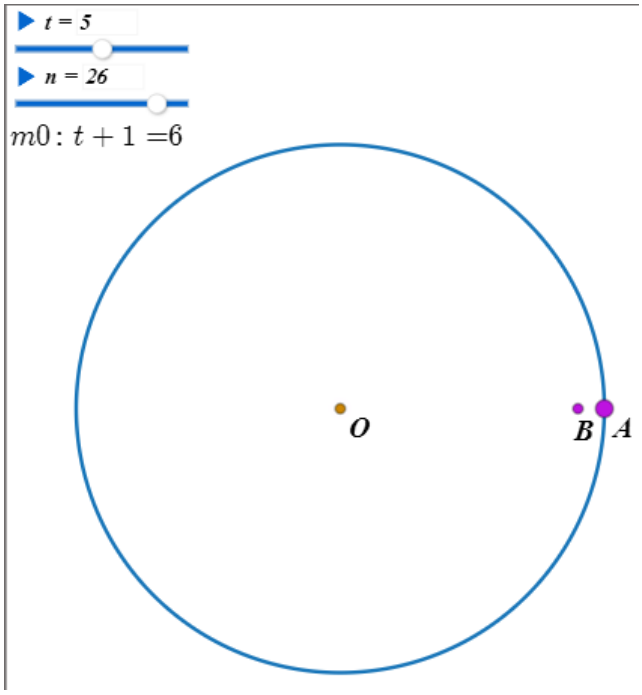


方法还有很多,但这三种方法是比较常见和实用的方法，现在我们来总结一下，我们迭

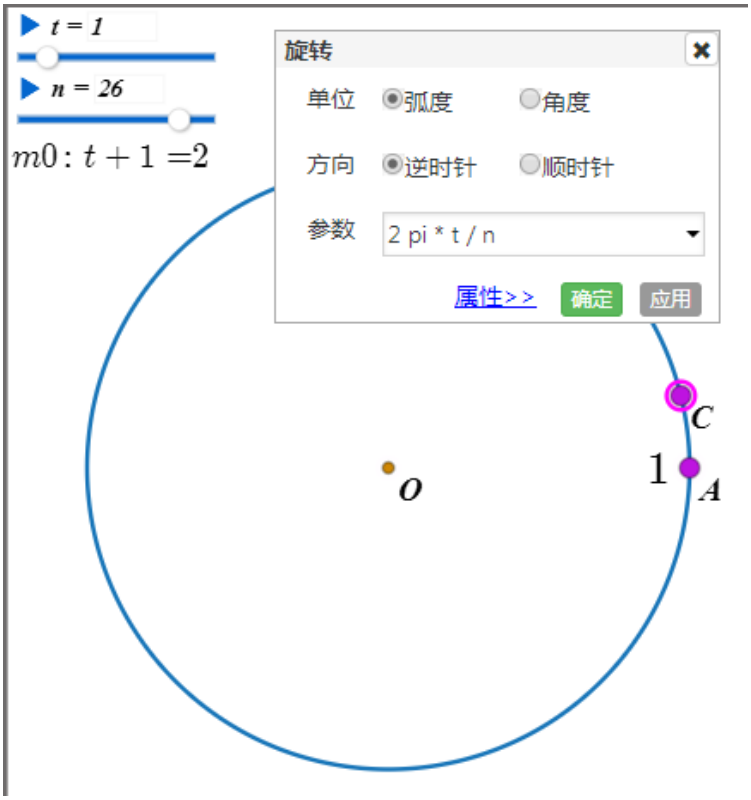
代时如何能够实现我们想要的效果。

现在再来看看下面这个做法为什么不成功.

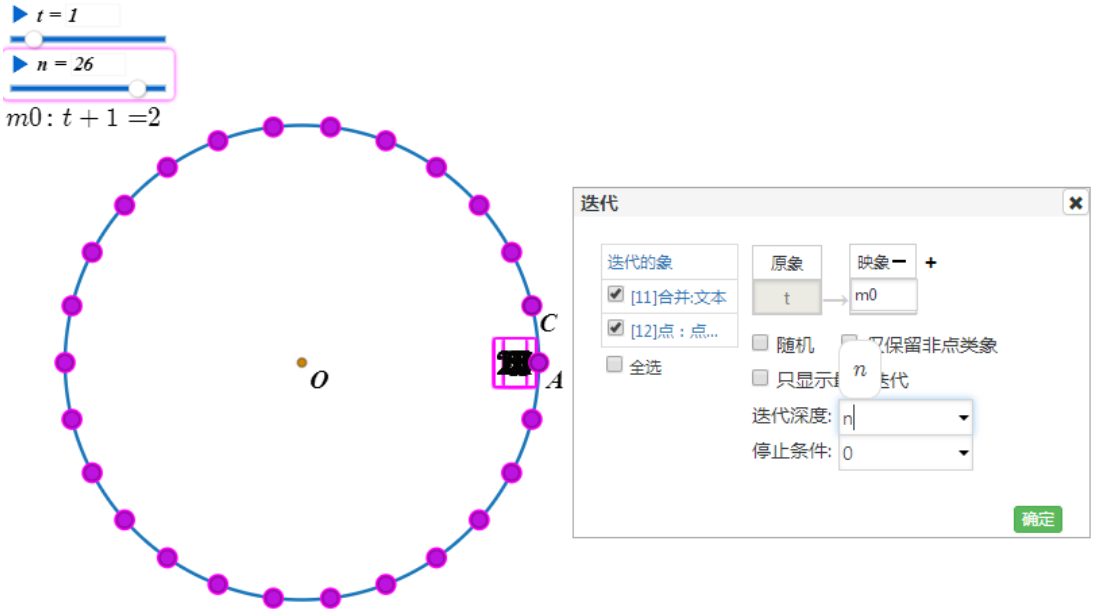
失败方法:



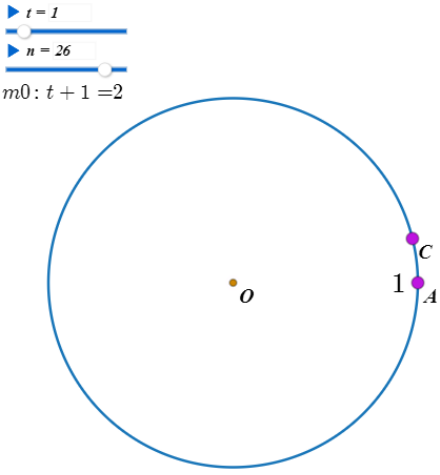
1.上图和前面步骤类似，点 A 是自由点，点 B 是缩放点 A 得到，写下来把合并 t 的值到点 B，然后旋转点 A 到点 C，旋转角度是 2pi*t/n，旋转中心为点 O，隐藏点 B



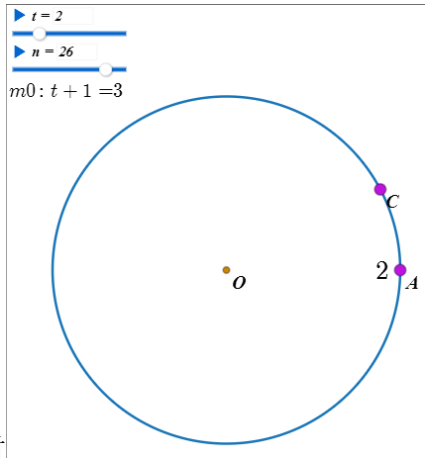
2.然后对变量 t 进行迭代，我们会发现结果是这样的



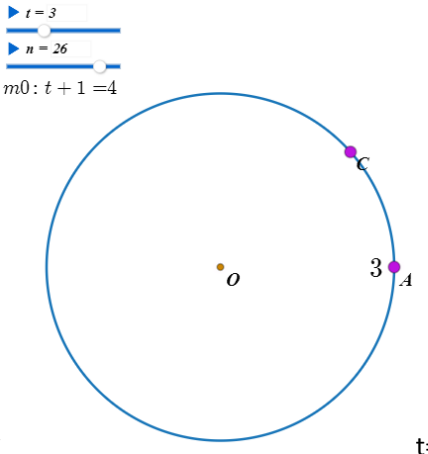
这里所有的标签都重合在一起，但是点还是分开来的，这是为什么呢？
接下来我们退到第 1 步的结果，连续变动变量 t 的值，我们会看到如下图的几种结果：



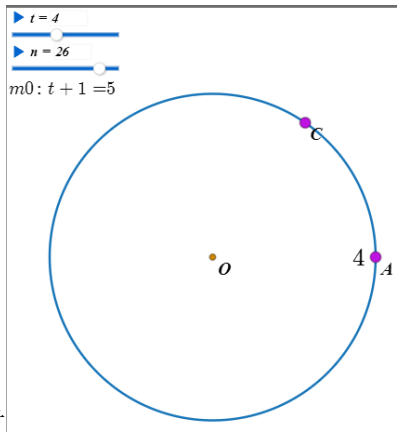
t=1 时



t=2 时



t=3 时



t=4 时

可见，但 t 不断变化时，标签的值时会改变的，点 C 的位置也是会改变的，但是标签的

位置却是不变的，这就是为什么迭代的时候可以得到点的图像，但是标签却重合在一起。那么到底这种办法为什么会失效呢？

首先我们来说一下网络画板迭代的时候“原象”，“映象”和“迭代的象”之间有什么关系？

1.原象：可以作为原象的对象有两种，一种是点（自由点、半自由点、非自由点都可以，但是迭代生成的点不行），第二种是变量或者计算表达式都可以。

2.映象：映象必须由原象得到，所以一定是原象的子对象，他们之间是父子关系。

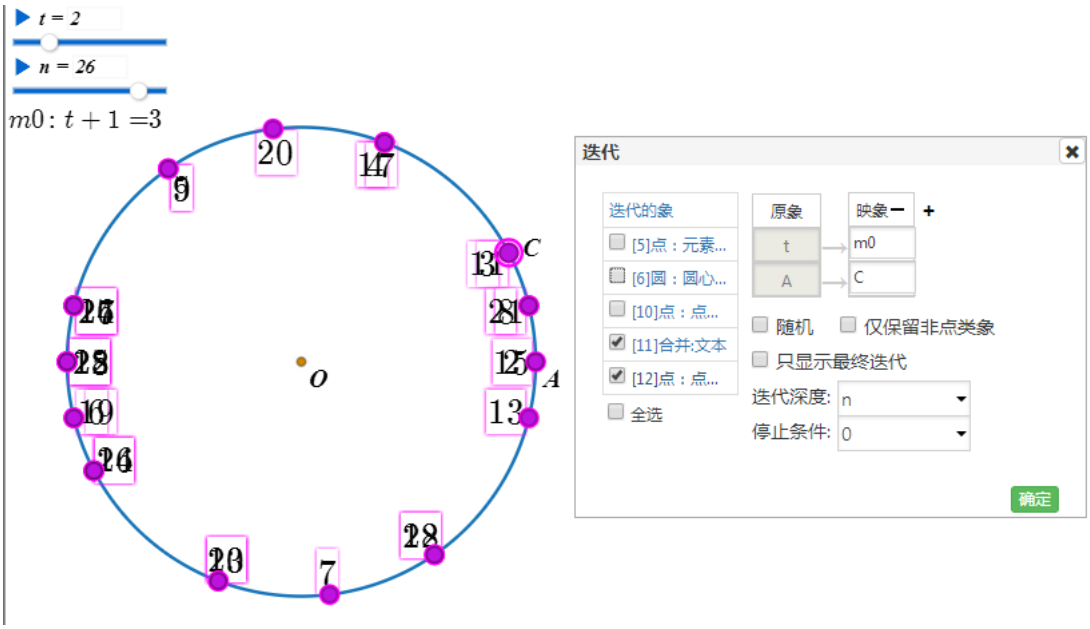
3.迭代的象：迭代的象就是可以生成迭代对象的对象，只有原象的子对象或者映象的子对象（当然，映象的子对象肯定也是原象的子对象，所以原象的子对象不限定为直接子对象，可能有多层子对象，甚至**包括原象生成的迭代象，这也是累次迭代的基础**）可以成为“迭代的象”，从而得到其迭代象。

接着，我们再来看看上面失败的做法，点 A 是自由对象，它不是原象 t 的子对象，所以不能通过对变量 t 的迭代来生成点 A 以及点 A 的子对象的迭代象。而标签 t 却绑定在点 A 的子对象 B 上，而对象 B 没有迭代象，所以绑定在点 B 上的标签是不会动的。但是因为标签是变量 t 合并到点 B 上的，所以标签就成为了变量 t 的子对象，所以在对变量 t 进行迭代式，标签的值是会发生变化的，也就是标签生成了对应的迭代象。再看点 C，它是由点 A 绕点 O 旋转 2pi*t/n 得到，所以点 C 就成为变量 t 的子对象，所以当对变量 t 进行迭代时，点 C 也可以生成迭代象，所以就得到圆周上的一些列点。

那么，我们应该如何修改迭代规则，是的标签不光内容会迭代，标签的位置也会随着 t 的变化而变换呢？

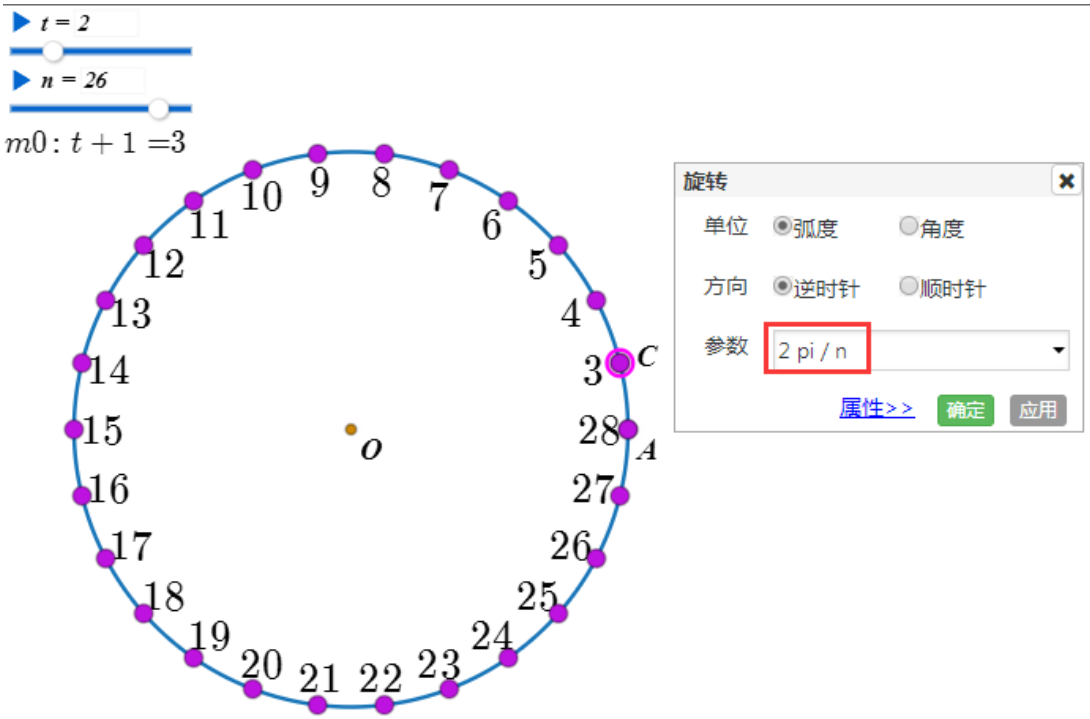
知道了迭代的原理，接下来可以这样修改。为了使得标签的位置跟随改变，所以标签绑定的对象点 B 必须在迭代中成为原象的子对象，而要做到这一点，只需要在迭代规则中增加点 A 为原象即可，迭代规则如下：

我们会发现



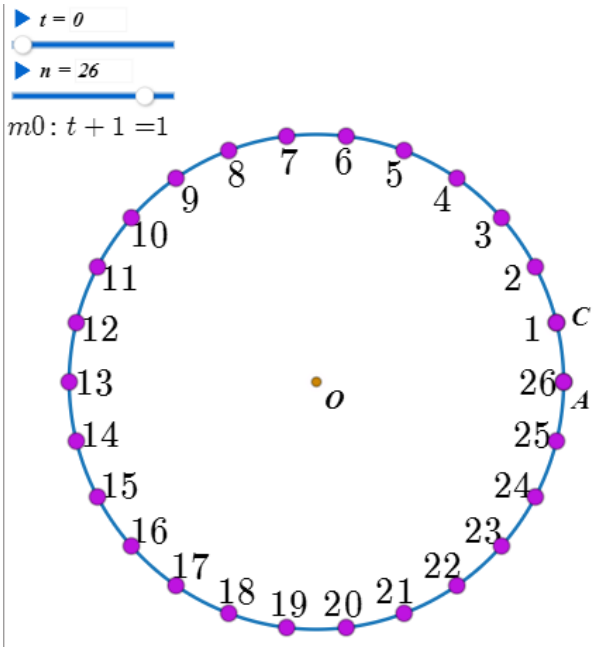
我们发现，标签的位置的确有变化了，但是为什么不是一个挨着一个呢？

要一个挨着一个的话，只需要把点 C 的旋转角度改成 2pi/n 即可。

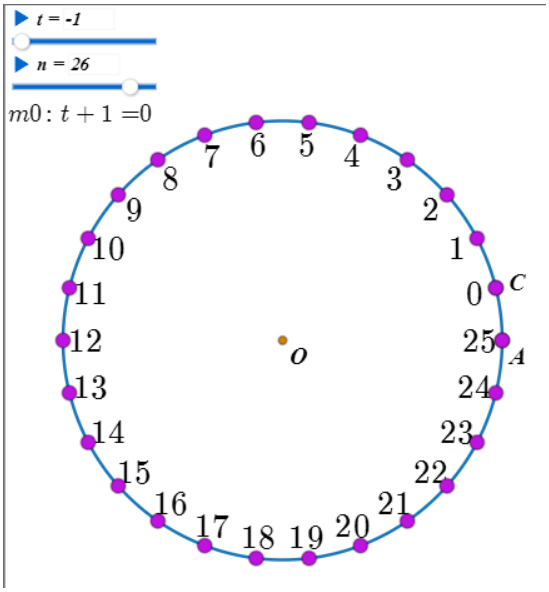


这是因为，如果不这么修改，点 C 的位置既要收到原象 t 的影响，又要受到原象 A 的影响，这两个影响共同叠加，就会造成这一现象。

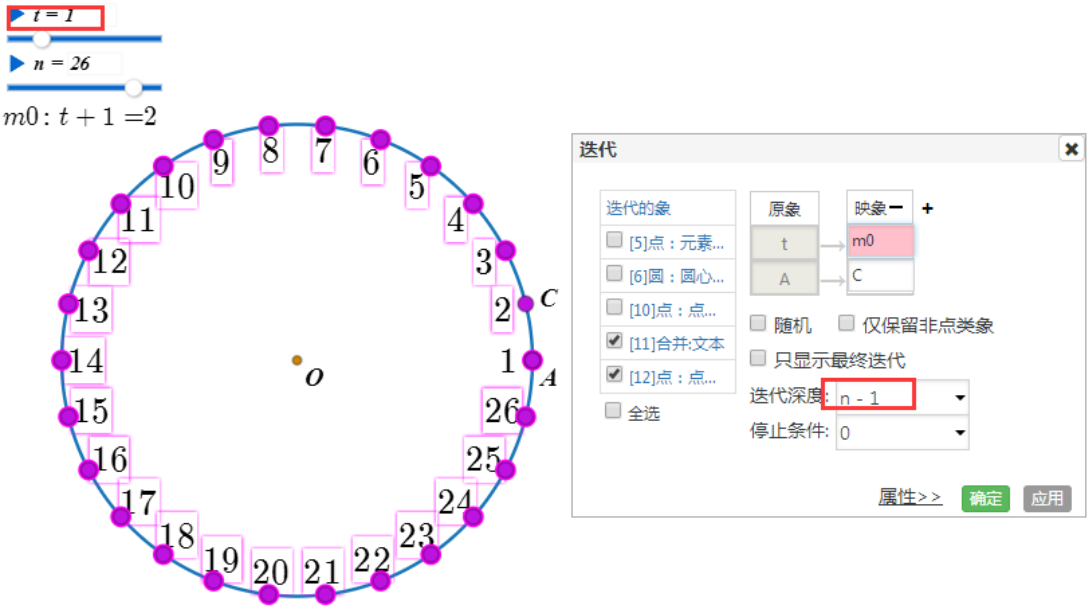
但是还有一个问题，但修改完后，把变量 t 拖回到 0 时，得到的结果如图：



标签为 1 的点跑到了点 A 上面一格，当把变量 t 变为-1 时，也不是我们想要的结果。



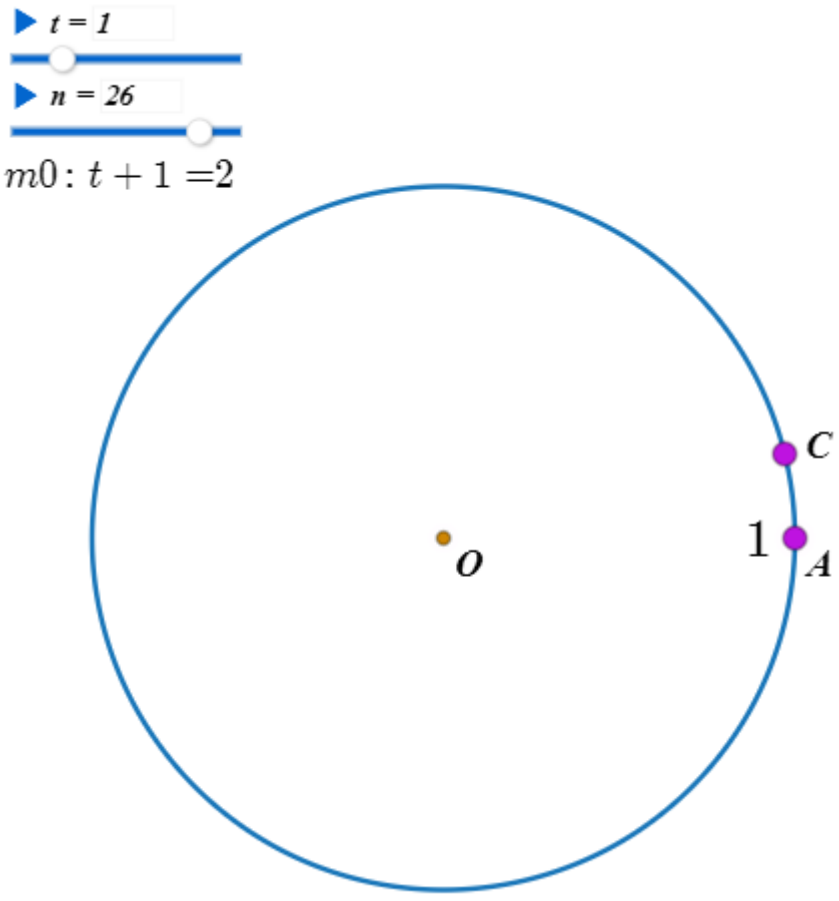
只有当我们把变量 t 变为 1 时，并且迭代深度改成 n-1 时，才达到了我们想要的效果。



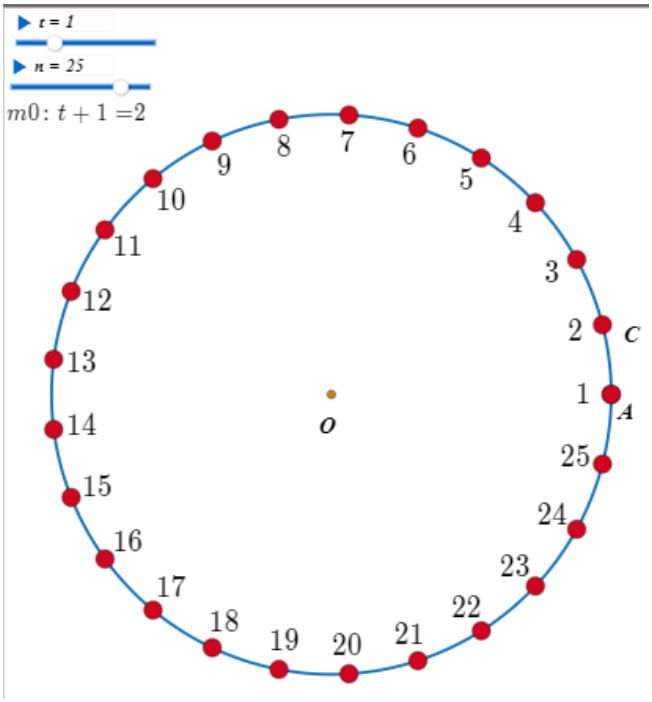
此时，标签 1 不是迭代象，标签 2 到 26 都是迭代象，也就是此时整个图像分为 2 类，一类是迭代前就存在的对象（比如标签 1），一类就是迭代象（比如标签 2 到标签 26）.

为什么必须要借助于迭代前的对象才可以达到效果？

这是因为我们迭代之前的状态是这样的，标签是在点 A 的位置，但你不想利用初始标签

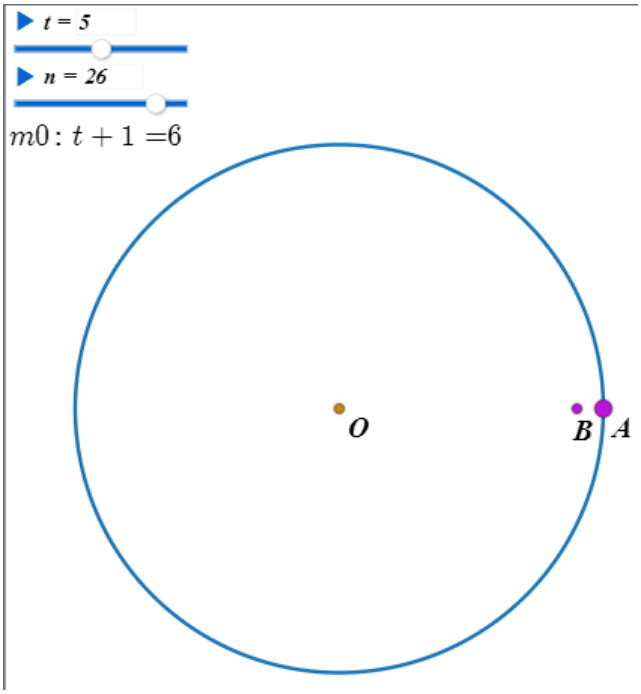


而想都利用迭代象的话，目前是做不到的。因为迭代象的第一个位置始终是从原始对象的下一个位置开始的。比如说在这里，原始对象就是标签 1 的位置，那么，当你进行迭代时，点 A 迭代到下一个位置点 C，那么标签 1 也会跟着到下一个位置，也就是点 C 的位置。所以生成的迭代象，都是从点 C 的位置开始的。



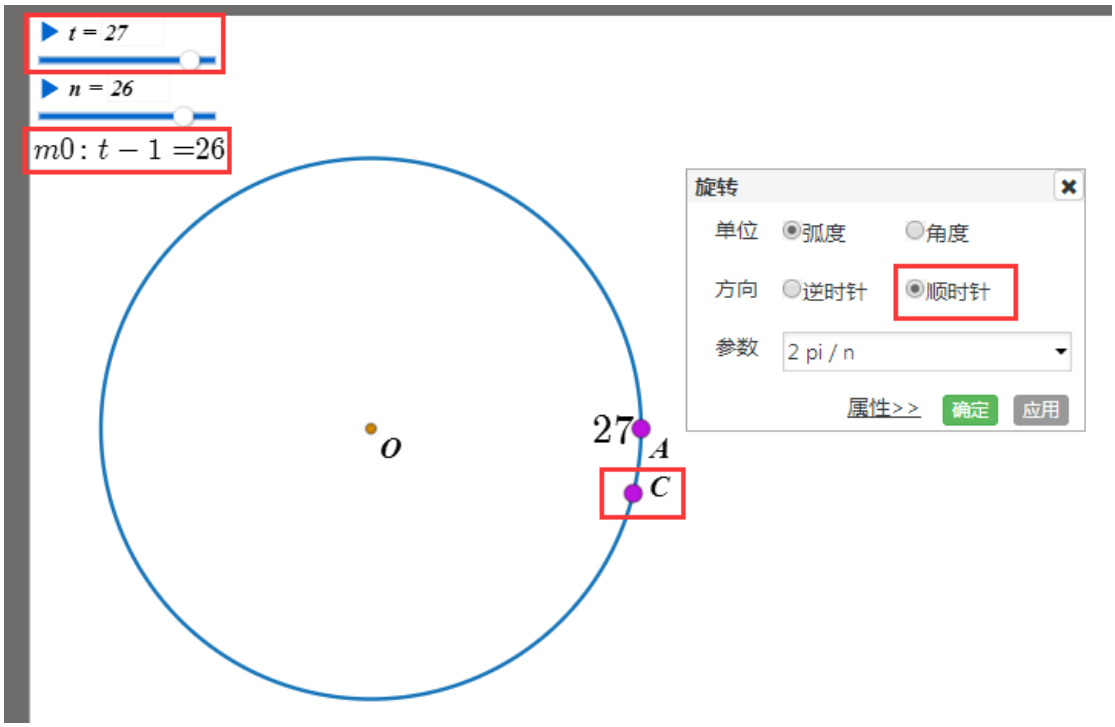
而要实现的图中，点 C 处的标签是 2，所以意味着它之前在点 A 处的标签必须是 1 才行，而当把变量 t 的值变为 1 时，点 C 以及点 C 后面的点都是正确的迭代标签，但是因为是从点 C 开始生成对应的迭代象，所以点 A 处就不是迭代象，而是初始的对象。

接下来我们再来探讨一下在一开始的这个状态下，能否实现全部图像都是由迭代象得到

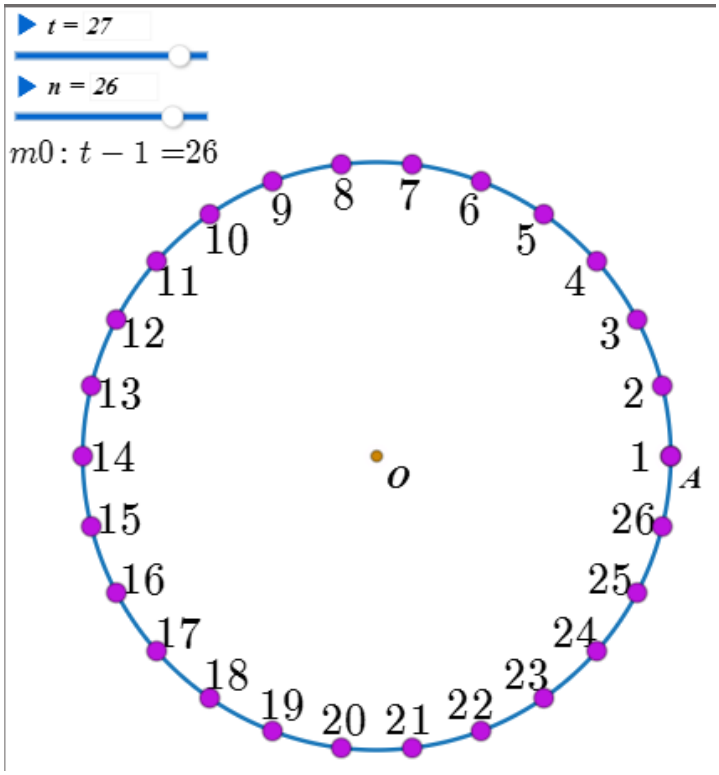


在这个状态下，当你把变量 t 合并到点 B 时，要使得标签位置变动，我们只能让点 A 成

为原象，但是我们说，迭代象的位置是从初始对象（在这里就是点 A）这个位置的下一个位置开始得到第一个迭代象，所以，我们只能使得点 A 的这个位置成为迭代象的最后一个位置（因为点 A 这个位置的下一个位置才是第一个迭代象的位置）。那么应该如何做到呢？可以这样做：



把点 C 的旋转方向改成顺时针，变量 t 的值从 27 开始，计算式 m0 变成 t-1 就行

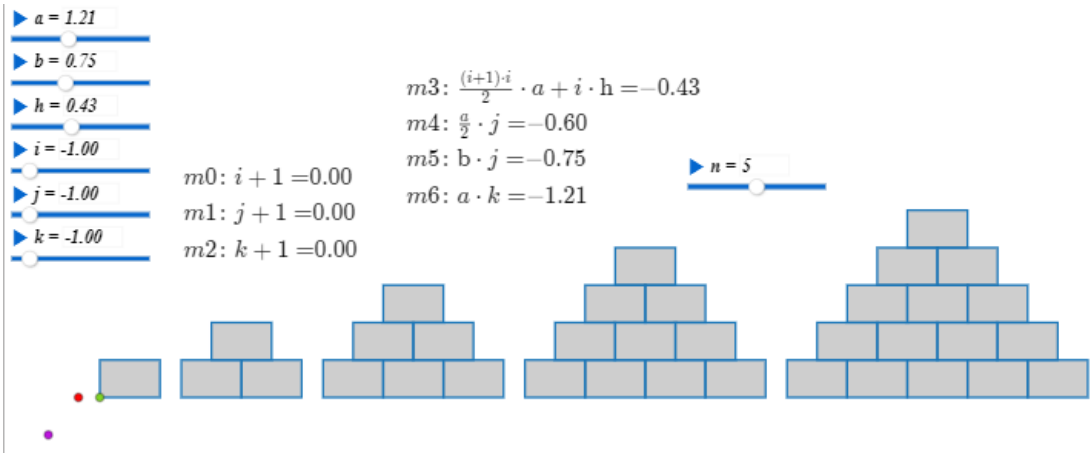


总结：从以上迭代过程可以看出，网络画板在迭代方面的确非常强大，迭代很自由，几

乎没什么限制，只要原象和映象之间存在父子关系，那就可以进行迭代，除非他们的子对象只包含计算值，那就不能生成迭代，其他都能够进行迭代。但是迭代时也要注意一些细节，就是你既可以只用变量进行迭代，也可以只用点进行迭代，当然也可以两者同时使用，这都可以根据你个人的需要来确定。还有一点就是，生成的迭代象的第一个位置都是从原始对象的下一个位置开始开始的。

第二部分　　累次迭代

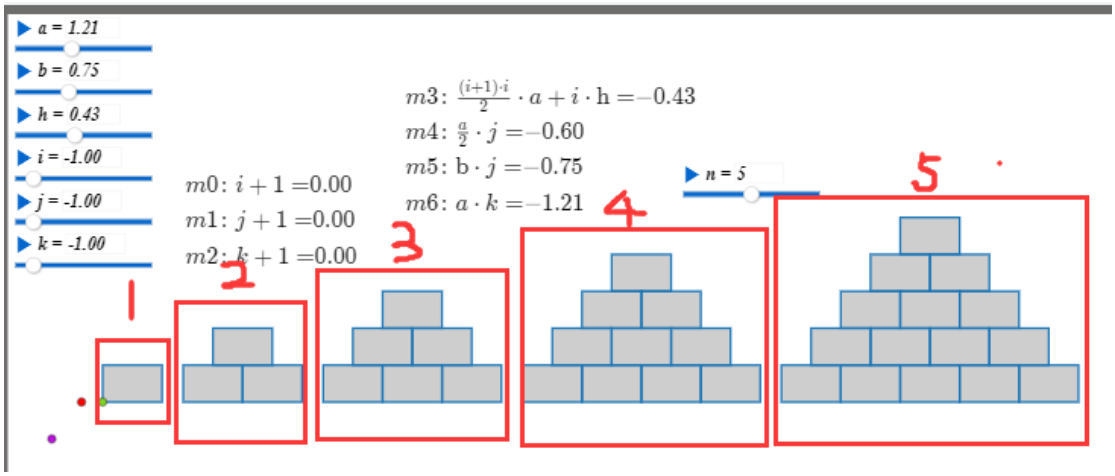
我们以一个例子来介绍累次迭代。下面一个例子是比较经典的累次迭代的例子，画板都可以做，但是相对来说用利用网板的累次迭代来实现比较简单，可以说几乎没有什么计算。



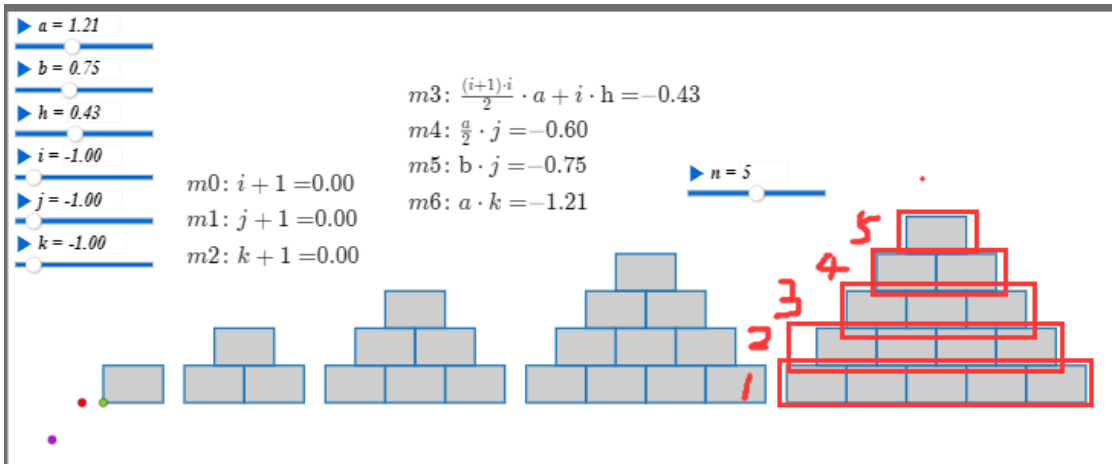
在上例中，变量 a 表示每一块砖的长，变量 b 表示每一块砖的宽，变量 h 表示每一堆砖块之间的间距，变量 i，j，k 分别表示累次迭代的最外层，中间层，最内层迭代的迭代变量。

接着我们来分析一下这张图的结构。

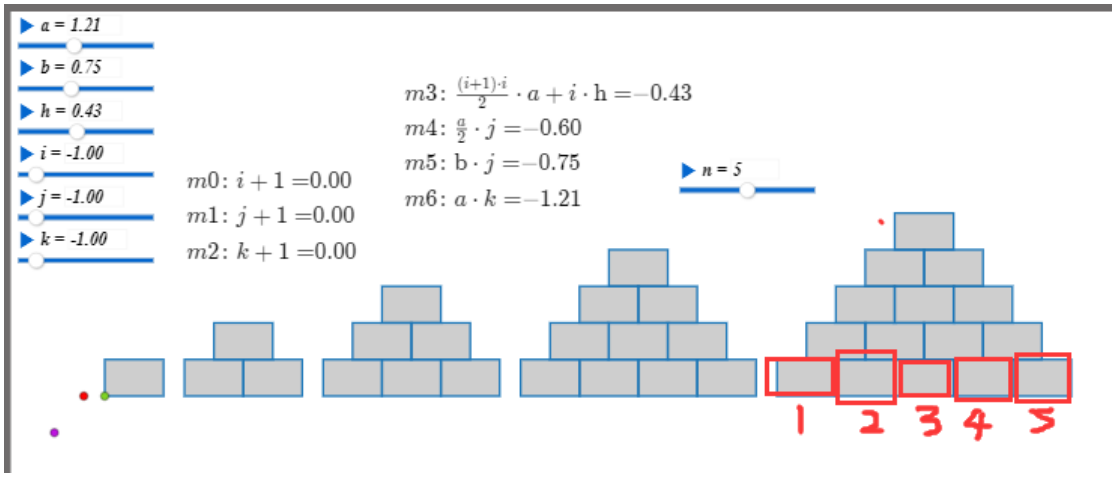
1.最外层可以看成有 i 堆：



2.某一堆又可以看成有 j 层：



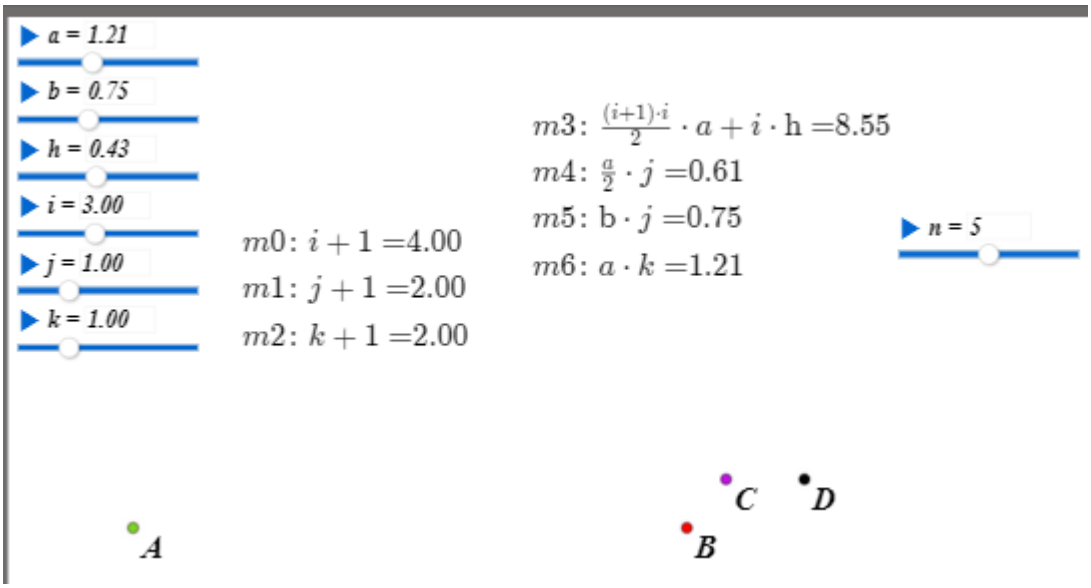
3.某一层有可以看成有 k 个：



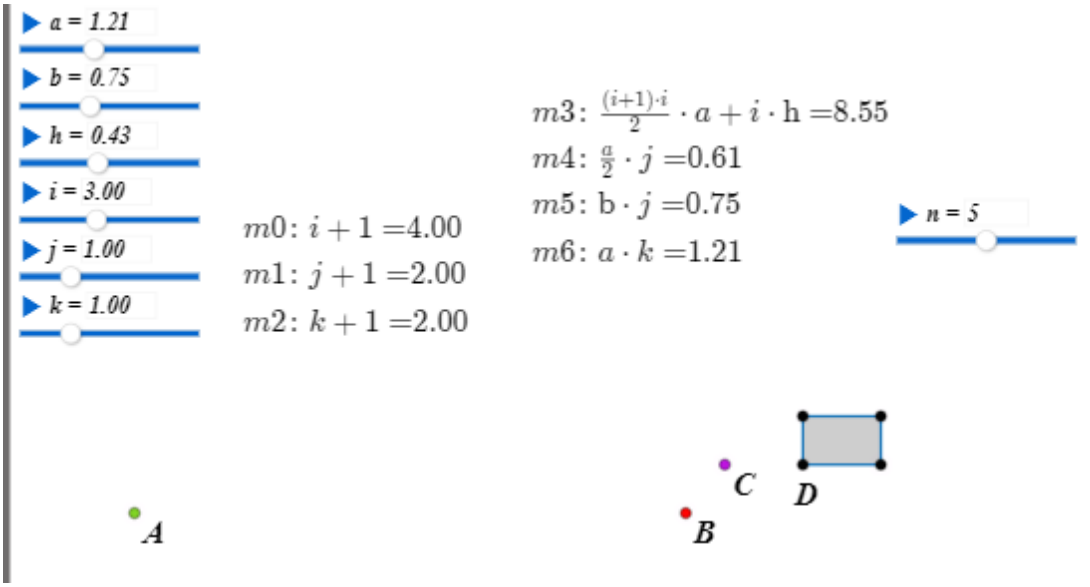
所以，可以考虑累次迭代，而且总共分 3 次迭代，并且迭代的时候需要先从内而外依次迭代。也就是说必须要先迭代生成某一层，然后再迭代生成某一堆，最后再迭代生成整个图像。但是，在第一部分中也说过，你要能产生迭代像，迭代的对象必须是原像的子对象，所以再真正迭代的时候我们必须反过来，先利用最外层的迭代变量 i 生成一个点 B，然后再利用这个点 B 再结合中间层的迭代变量 j 再生成一个点 C，最后利用这个点 C 再结合最内层迭代变量 k 来生成一个点 D。这样，就形成了**一个父子链：变量 i→点 B（由变量 i 控制）→点 C（由变量 j 控制）→点 D（由变量 k 控制）**。接着，只要用变量 k 作为原象对点 D 进行迭代生成迭代像 1（即某一层），然后再用变量 j 作为原象对迭代像 1 进行迭代生成迭代像 2（即某一堆），最后再利用变量 i 作为原象对迭代像 2 进行迭代生成迭代像 3（即全部图像）。

具体步骤如图所示：

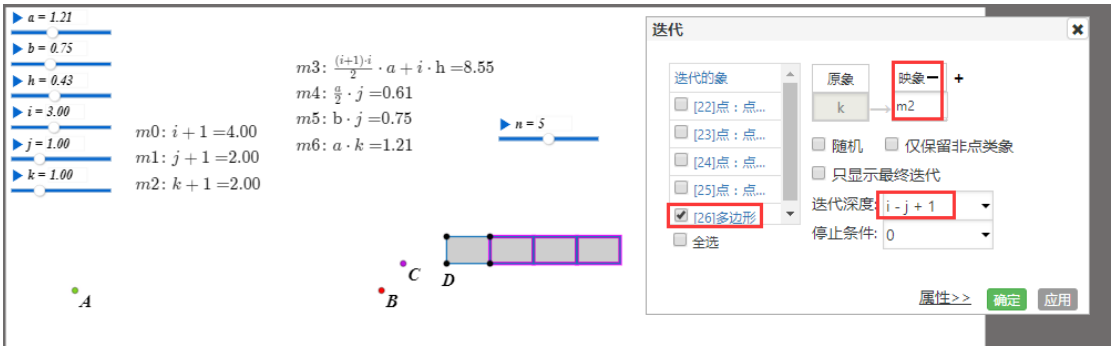
1.创建一个自由点 A，用于控制整个图形的位置，再把点 A 向右平移 m3 个单位得到点 B；再把点 B 向右平移 m4 个单位，向上平移 m5 个单位得到点 C；最后把点 C 向右平移 m6 个单位得到点 D



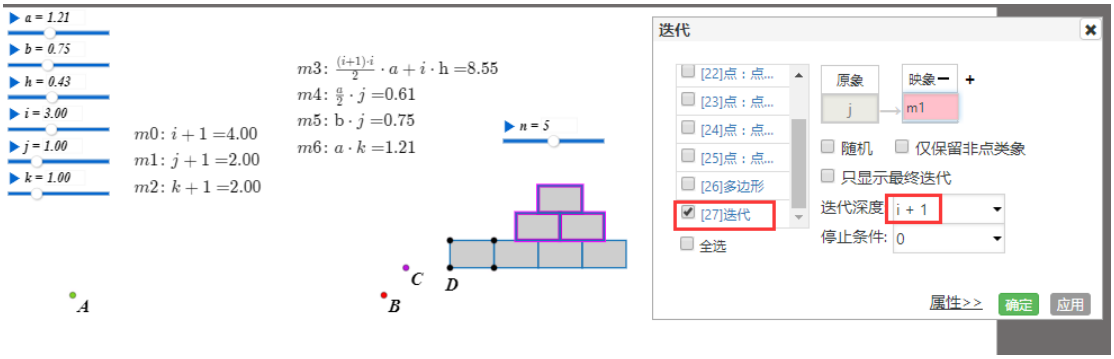
2.把点 D 分别向右平移 a 个单位，向上平移 b 个单位，向右上方平移（a,b）个单位，以这四个点为顶点构造矩形



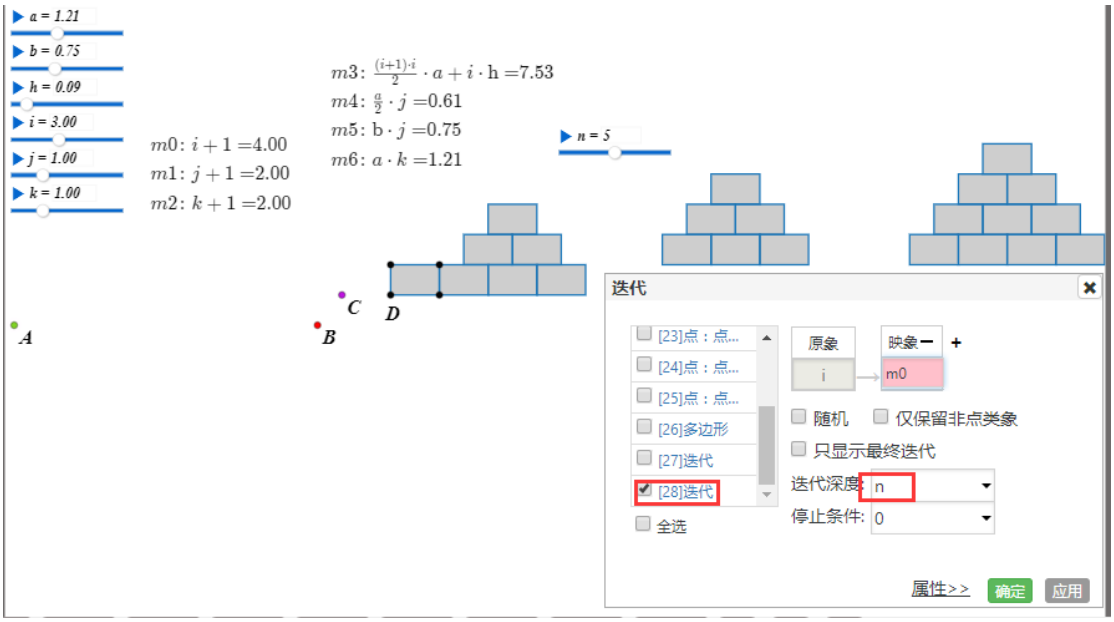
3.先对变量 k 作为原象进行迭代，迭代深度为 i-j+1，迭代的象为构造的矩形



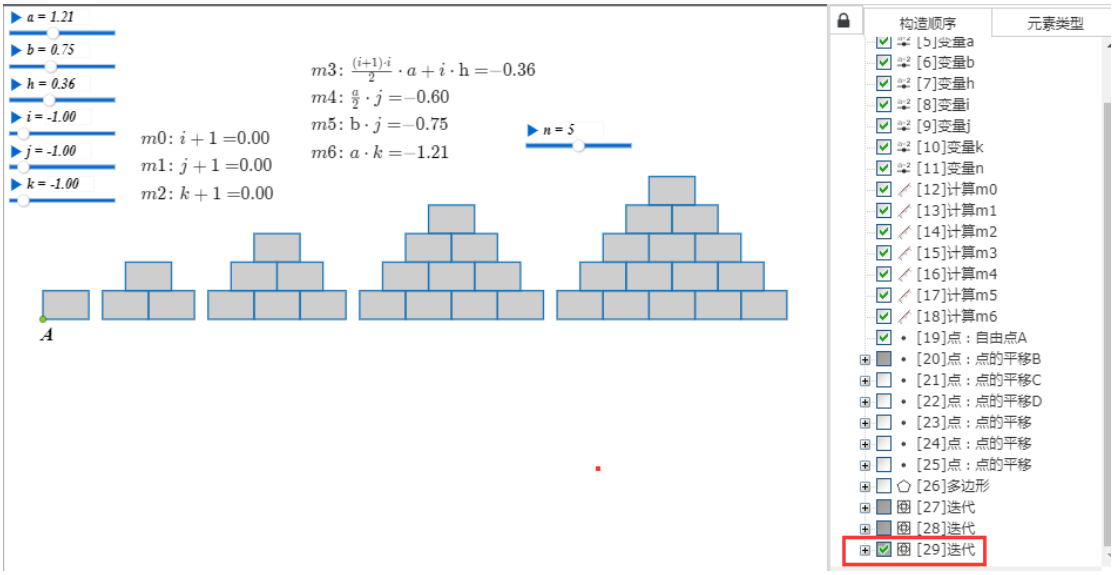
4.再对变量 j 作为原象进行迭代，迭代深度为 i+1，迭代的象为之前的迭代像



5.再对变量 i 作为原象进行迭代，迭代深度是 n，迭代的象为最近一次生成的迭代像

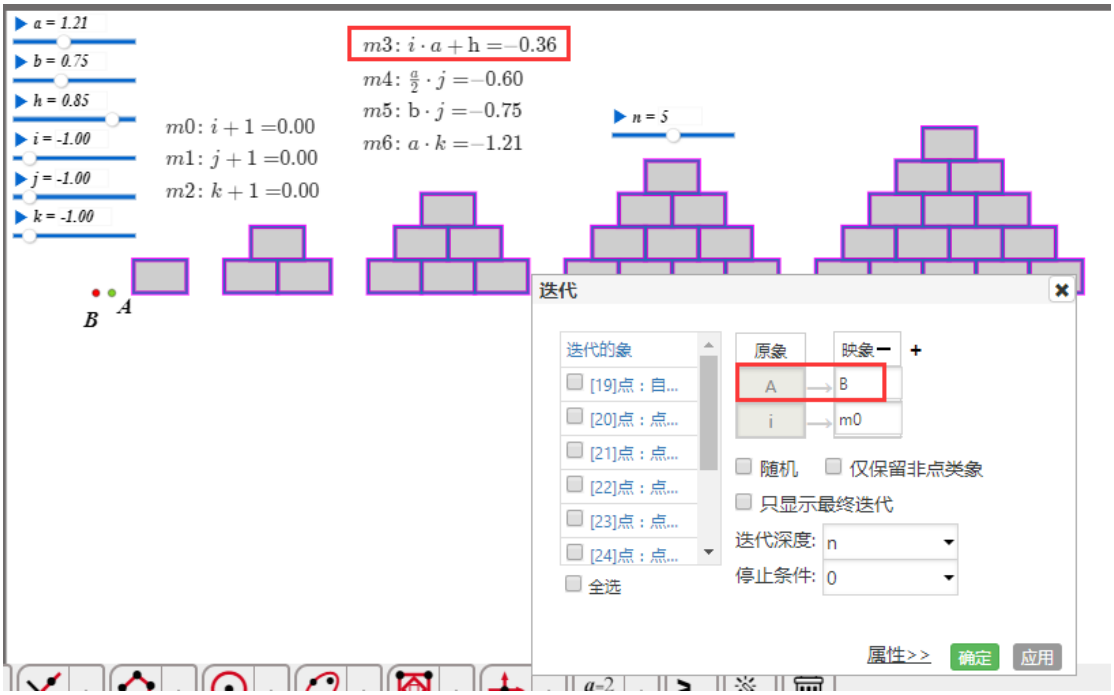


6.最后调整所有变量的初始值：i，j，k 的初始值都是-1，a，b，h 的初始值随意，隐藏不必要的对象，保留最后一次的迭代像。



这个方法有个小问题，就是在计算每一堆之间的间隔的时候，计算式 m3 相对来说有点

复杂，当然其实也不难理解，因为每一堆随着 i 值的增加，最底层的砖块数量从 1+2+3+……+n，所以才会有 m3 的表达式。那么，能不能把 m3 的表示式化的更简单些呢？答案是肯定的。



不过这里有个缺点，就是整个图形的左下角不能和点 A 始终重合，不像上面的方法图形的左下角始终和点 A 重合。不知道大家有没有办法解决这个问题？

总结：要实现累次迭代，主要是要生成一个形如这样的一个父子链：

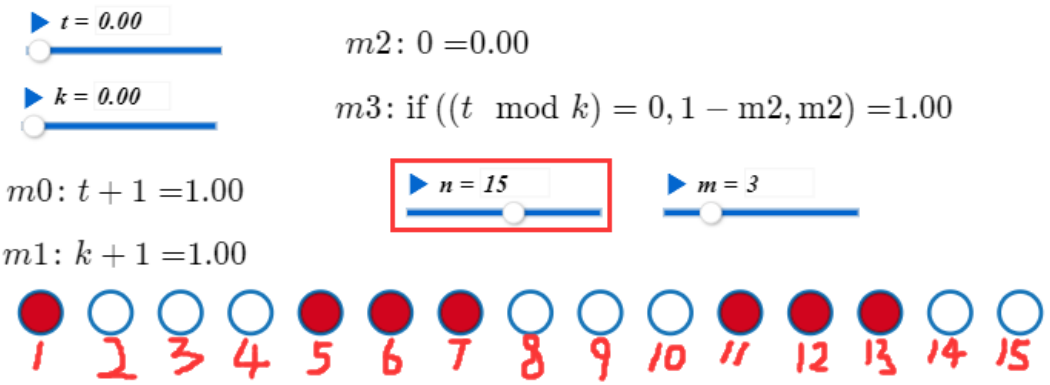
变量 i→点 B（由变量 i 控制）→点 C（由变量 j 控制）→点 D（由变量 k 控制）

迭代变量 i 生成迭代像 2 的迭代像 3

迭代变量 j 生成迭代像 1 的迭代像 2 ← 迭代变量 k 生成迭代像 1

第三部分　网板 83 期打擂题制作分享

接下来我们应用累次迭代的思想来讲解网络画板赛第 83 期打擂题的制作方法。



网络画板赛 83 期打擂题要实现的效果是：有 n 个灯泡并列的排成一排，每个灯泡对应编号从 1 到 n，一开始每个灯泡都关闭，接着拉第 1 次开关，所有灯泡编号能整除 1 的都改变原来的开关状态（即原来开的变关，原来关的变开），所以第 1 次开关的时候，因为所有灯泡的编号都能整除 1，所以所有灯泡都改变状态，而原来这些灯泡都是关着的，所以拉第 1 次开关的时候，所有灯泡全部点亮。接着拉第 2 次开关，还是所有灯泡编号能整除 2 的都改变原来的开关状态，接着拉第 3 次开关，还是所有灯泡编号能整除 3 的都改变原来的开关状态，……如此操作，一直到拉第 n 次开关，因为此时只有第 n 个灯泡的编号能够被 n 整除，所有只有第 n 个灯泡的的状态改变。至此，操作结束。

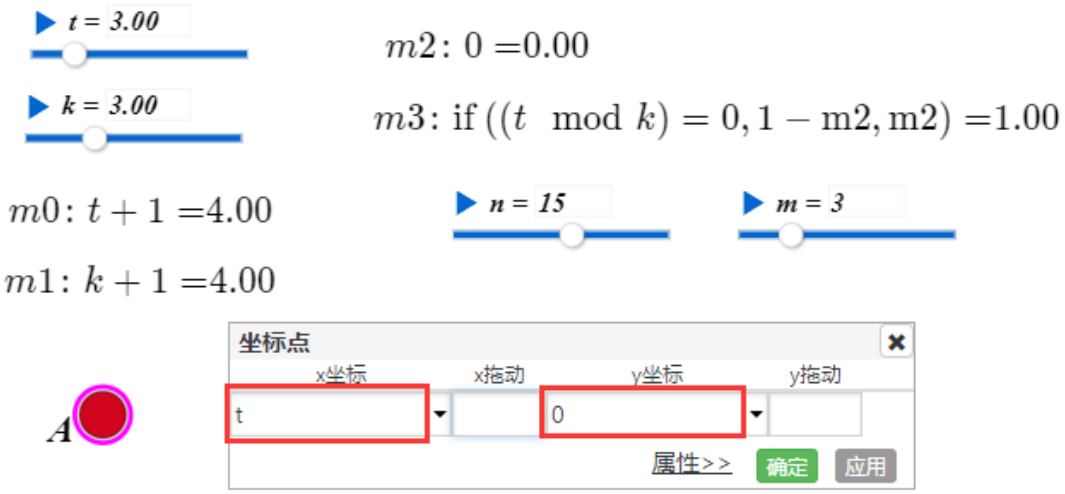
首先，这 n 个灯泡肯定要迭代得到的，如果不需要用到灯泡的编号，那么迭代的时候只用点来作为原象就可以生成这 n 个灯泡的迭代像。但是根据题目意思，每个灯泡的开关状态和这个灯泡的编号有关，所有生成这 n 个灯泡的迭代像必须要用变量作为原象来进行迭代。在这里，我用变量 t 来控制生成 n 个灯泡的迭代像。

接着，最难的问题来了，要生成 n 个灯泡的迭代像很容易，但是如何能够让不同编号的灯泡的开关状态用拉开关的次数（在这里我用变量 m 表示）控制呢？

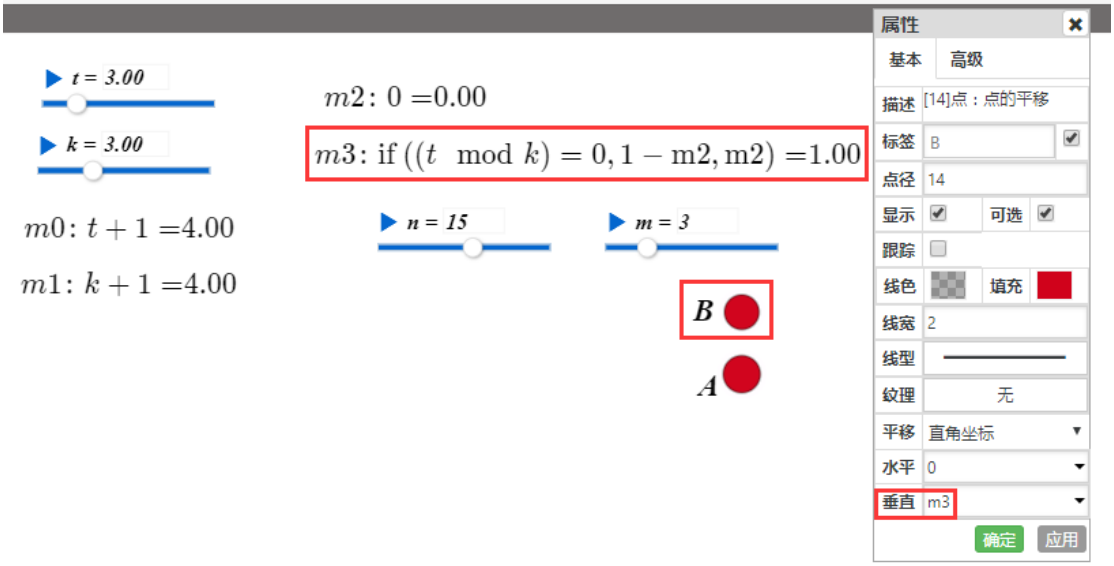
我们可以这样构造一个父子链：**由变量 t→点 A（由变量 t 控制）→点 B（由变量 k 控制）→迭代像（取最终迭代像）**

按照这个思路，

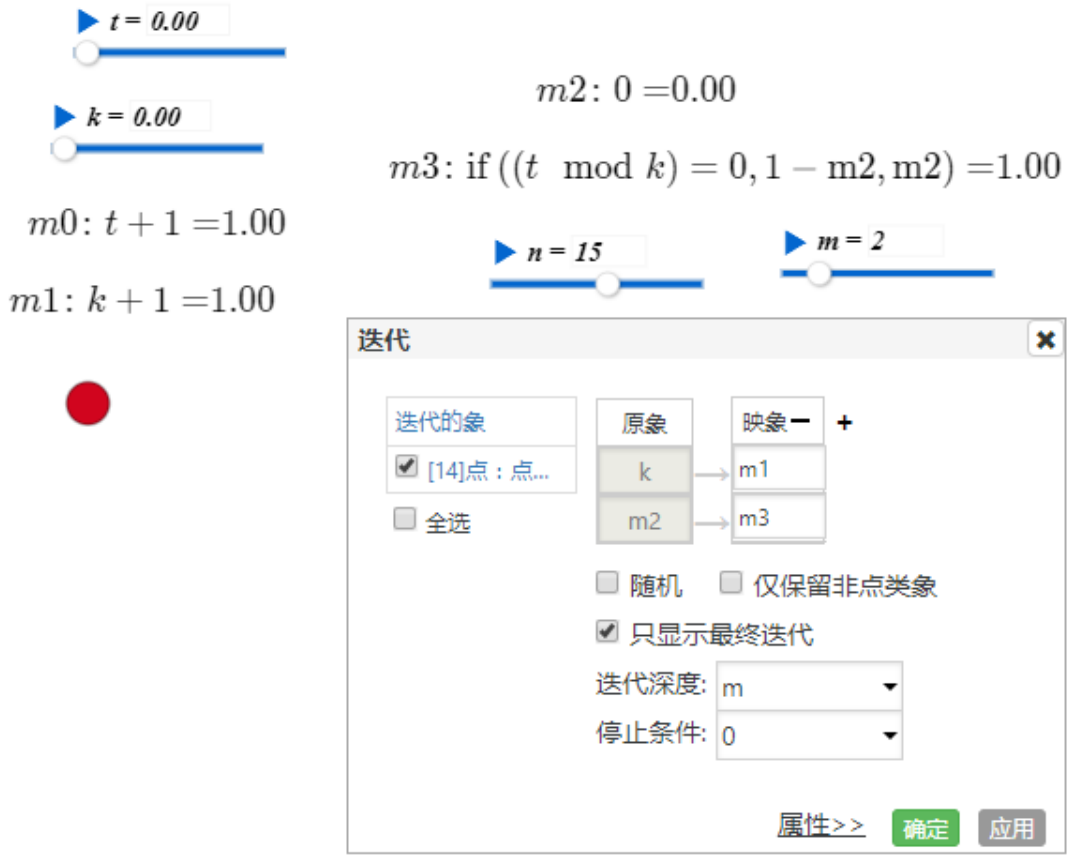
1.我们先用变量 t 绘制点 A



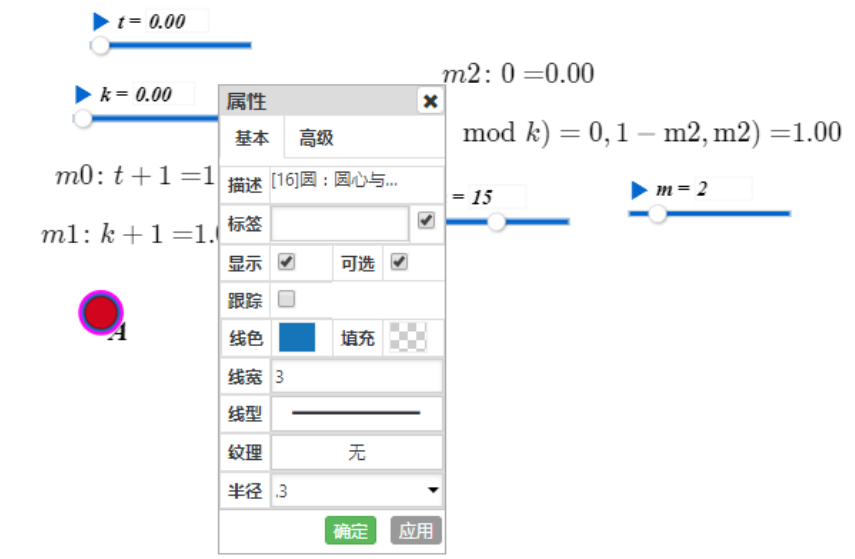
2.接着就是利用点 A 和变量 k 来控制生成点 B，那么我们应该如何做呢？如何做要取决于我们想要让点 B 实现怎么样的效果。在这里，点 B 的作用是用来控制某个编号的灯泡的显隐的，而灯泡的显隐可以用点的两种效果来代替：一种是点存在，一种是点不存在。所以在绘制点 B 的时候，只要平移点 A，如果在这个位置上的灯泡是点亮的，那么就把点 A 平移（0，0）的位置（也就是原地不动，但是该点还是存在的），如果在这个位置上的灯泡是关闭的，那么就把点 A 平移一个无穷大，比如（0，∞）。



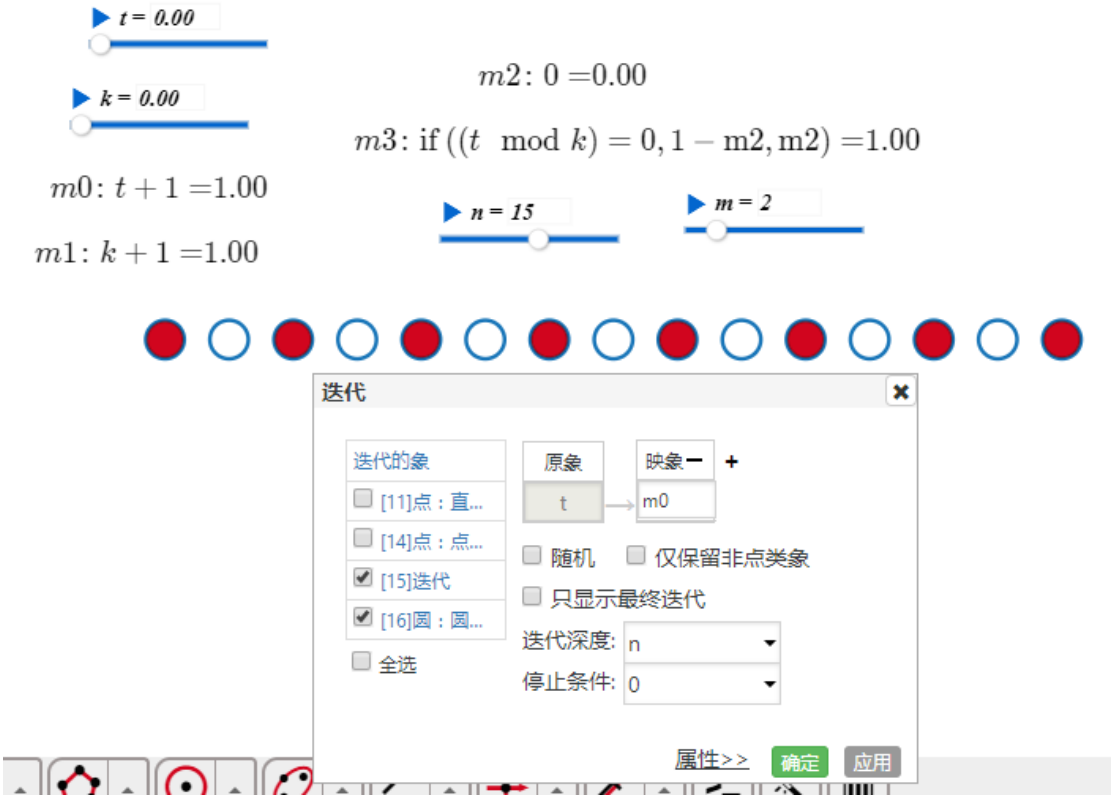
3.对变量 k 进行迭代，生成点 B 的最终迭代像



4.以点 A 为圆心，0.3 个单位为半径作圆



5.对变量 t 进行迭代，生成点 B 的最终迭代像的迭代像，并隐藏原来的点和圆



至此，本赛题制作完毕。

但是，还有几个问题需要回答：

1.在生成点 B 的迭代像的时候除了变量 k→k+1，为什么还要 m2→m3？

答：这是因为，某处灯泡的状态，但从拉第 m 次到拉第 m+1 次，第 m+1 次的灯泡的状态是和第 m 次灯泡的状态直接相关的，也就是说，在决定拉第 m+1 次开关时该灯泡的状态是需要知道它前面一次灯泡的状态的。所以，我们可以用 m2 的值来表示该灯泡上一次的状

态，用 m3 的值来表示灯泡下一次的状态。具体如何决定，就取决于计算式 m3 的表达式，该表达式是这样的：if(mod(t, k) == 0, 1 - m2, m2)。它表示，当拉第 k 次时，计算编号为 t 的灯泡的编号 t 是否能被 k 整除，如果能，则要改变灯泡的状态，所以用结果 1-m2 来表示（即当原来的 m2 的值为 0 时，其结果变为 1-0=1；当原来的 m2 的值为 1 时，其结果变为 1-1=0），这样就达到了改变灯泡状态的目的。而当 t 不能整除 k 时，则结果不变，所以 m3 的值还是 m2。

由此可见，由 m2→m3 取决于该问题的要求。**当我们需要利用上一次的结果来计算下一次的结果的时候，那么就必须使用迭代了。**

2.为什么平移点 B 的时候是（0,0/m3），而不是（0,0/m2）呢？

答：刚开始是觉得把点 B 平移（0，0/m2）的确更合理。因为当我们把变量 t 和 k 都拖动到初始值（t=0，k=0）时，m2=0，m3=1。这正好符合我们的要求。因为对于点 B 来说，它平移（0，0/m2）时，初始值为（0，∞），表示这个点不存在，正好符合开关次数 m=0 时，每个灯泡都关闭的要求。当然，m=0 时，灯泡都关闭和点 B 平移得到的点的初始状态没有关系，因为这 n 个灯泡最终是对迭代像进行迭代而得到的图形，而当 m=0 时，表示**迭代次数为 0，此时表示迭代像不存在，当对不存在的迭代像继续迭代时，当然也不存在的**。所以这个是正确的。接下来我们看 m3=1，这个表示的是，迭代次数 m=1 是灯泡的状态（m3=1 表示平移的点存在，所以就表示这个灯泡状态是点亮）。而这也是符合我们的预期的，就是当拉第 1 次的时候（m=1 时），所有灯泡都点亮。所以按理说把点 B 平移（0,0/m2）是对的。但是当我们试验一下就会发现，当 m=0，1 时，全部灯泡都是关闭的，当 m=2 时，全部灯泡都点亮了，当 m=3 时，所有偶数编号的等都关闭了，……。我们会发现，这和预期的结果是不一样的，而且也能够发现，这个结果和把点 B 平移（0,0/m3）的结果就只相错了一位，即当 m=2 时，把点 B 平移（0,0/m2）的结果正好是当 m=1 时，把点 B 平移（0,0/m3）的结果。

这原因其实就是因为在迭代中，m2 是原象，m3 是 m2 的映象，**当初始值不变的情况下，迭代原像产生的第 m 次结果就是迭代映像产生的第 m-1 次结果。**所以才会出现这样的现象。

但是，为什么实际的效果不符合我们的预期呢？其实原因是这样的，虽然看着初始值 m2=0，m3=1，其实我们需要把 t 的初始值从 1 开始，这样你会发现，此时 m2=0，m3=0 了。而我们对 t 进行迭代所产生的所有灯泡的编号 t 就是从 1 开始到 n 的。这样，如果把点 B 平移（0,0/m2）的话，点 B 的第一个迭代像的结果就是 m3 的值，此时 m3=0，就表示该点不存在，从而得到全部灯泡都是关闭的。这与拉第 1 次开关时灯泡全部点亮不一致，而把点 B 平移（0,0/m3）时，就出现正确的结果了。这个问题我感觉说得不是很清楚，但愿读者大概能懂我的意思吧。

最后，我们改编一下这个题目，看看如何实现，从而希望对累次迭代有更深入的体会。我现在还是有 n 个灯泡排成一列，现在拉第 m 次，如果 m 与灯泡的编号互质，则该处灯泡点亮，否则该处的灯泡关闭，该如何实现？

具体实现请看网址：http://www.netpad.net.cn/svg.html#posts/60625

现在有个问题回答一下，在这个问题中，需不需要用累次迭代？答案是：不需要。为什么呢？因为灯泡的状态只由变量 t 和变量 m 决定，而且最关键的是，灯泡的状态和它

之前的状态没有关系，编号为 t 的灯泡的状态直接可以用变量 t 和变量 m 的表达式得到，所以生成编号为 t 的灯泡的状态不需要进行迭代了。

结束语：虽然本文构思了近一个星期，但是由于本人对迭代理解的也不是很深入，所以很多东西在写之前还是比较模糊，甚至写完这篇文章后，也不敢保证我所理解的都是正确的，所以如果文章中有错误之处也在所难免，希望大家多多包涵，也希望大家能多提意见，多交流。

本人 QQ 号：33042626
网络画板群号：307450983