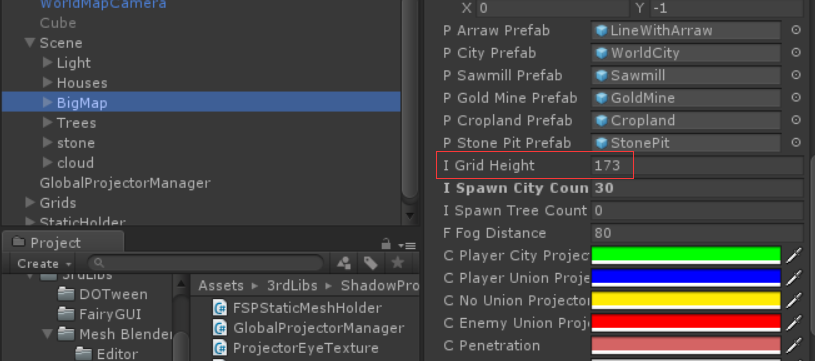
1. Terrain Coordinate

1.1 坐标系统

地表有两套2维坐标，一套是OddQ格子坐标。一套是UV坐标。Uv=(0,0)表示地面的左下角，(1,1)表示平面的右上角。所有对OddQ格子的处理，基本上全部是先转换成UV，再处理。

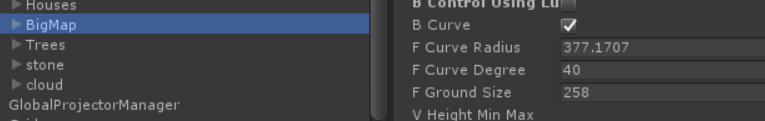
在代码中，已经假设了地面是正方形的。所以OddQ格子与UV的对应关系，由这个Grid Height参数唯一决定：



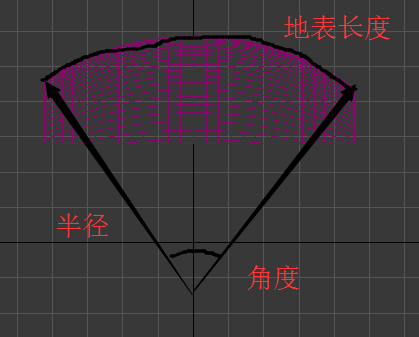
比如Grid Height=173。表示地面会分成200x173个OddQ格子。

1.2 曲面

曲面地表需要勾选bCurve，并且填写这几个参数：



它们的含义如图：



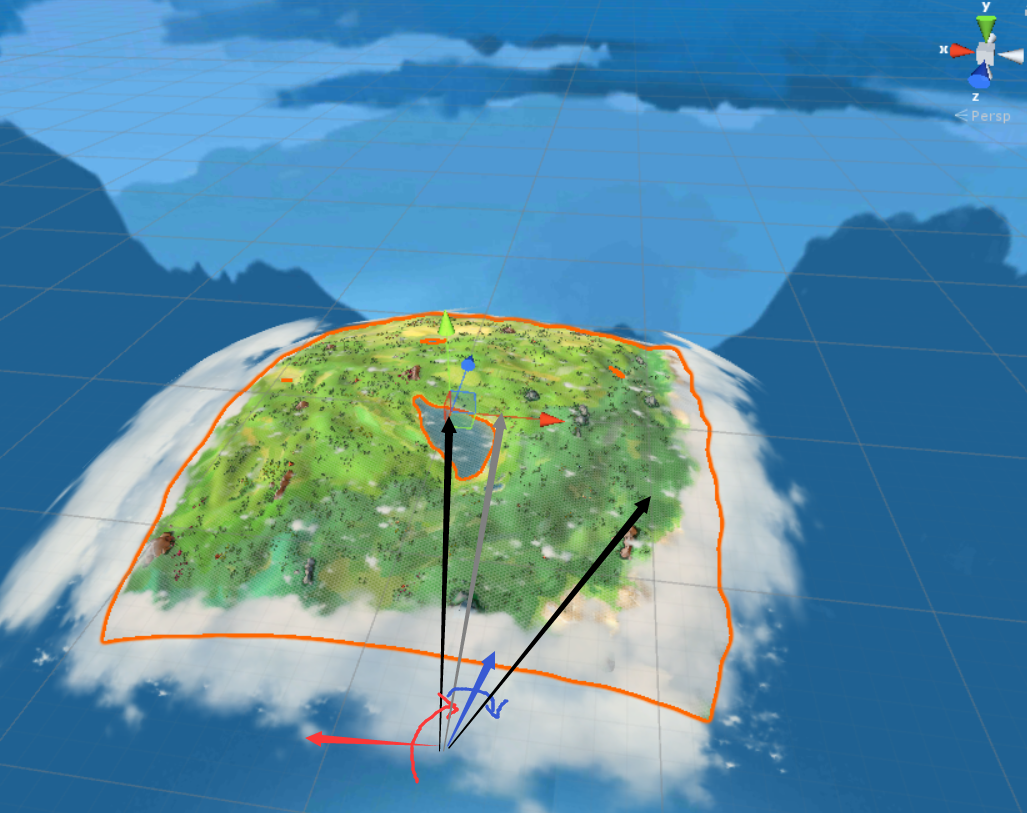
377\*pi\*40/180和258并不严格相等，原因是这些数值是之前的terrain转mesh时计算的。现在的版本这些数值大致相等就可以了。建议ground size（地表长度）略小于半径x角度。

坐标系统

假定uv=(u,v)

对应的地表位置为：

先由圆心向上的矢量沿着x轴旋转(u-0.5)\*40度。然后沿着-z轴喜欢转(v-0.5)\*40度指向的方向，然后做raycast。



1.3 平面

平面只需要去掉勾选m\_bCurve就可以了。

其它参数，是根据地表mesh的bound box计算出来的。

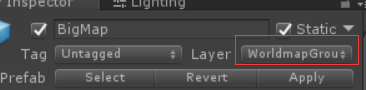
平面的坐标系统比较朴素。就是x=ground size \*(u -0.5), x=ground size \*(v-0.5)，然后做raycast。

1.4 其它设置

1. 需要加一个mesh collider



1. 需要设定layer为worldmap ground



1. 需要拖入地表prefab



1. 需要拖入行军线prefab



1. Fog distance



他决定了多远的物件会被影藏。

1.5 测试设置

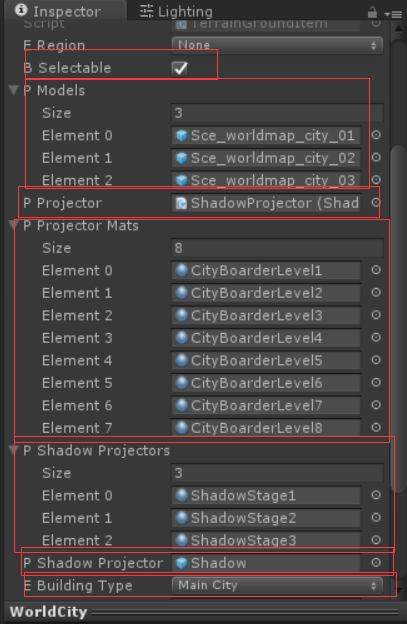
如果不勾选control using lua，可以直接在场景运行，做一些测试。



这两个参数可以随机造一些城市和资源建筑以供测试。

1. Terrain Ground Item

城市：



说明：

（1）selectable和building type

selectable其实表示可以被选中显示边界的物件。目前可以认为是城市就要勾。

此外building type必须对应。

（2）模型

模型分别对应于各个级别的模型。

（3）

关于projector我建议优化。这里是目前需要填入的：

目前对于城市必须指定projector（显示边界用的）以及shadow projector（显示影子）

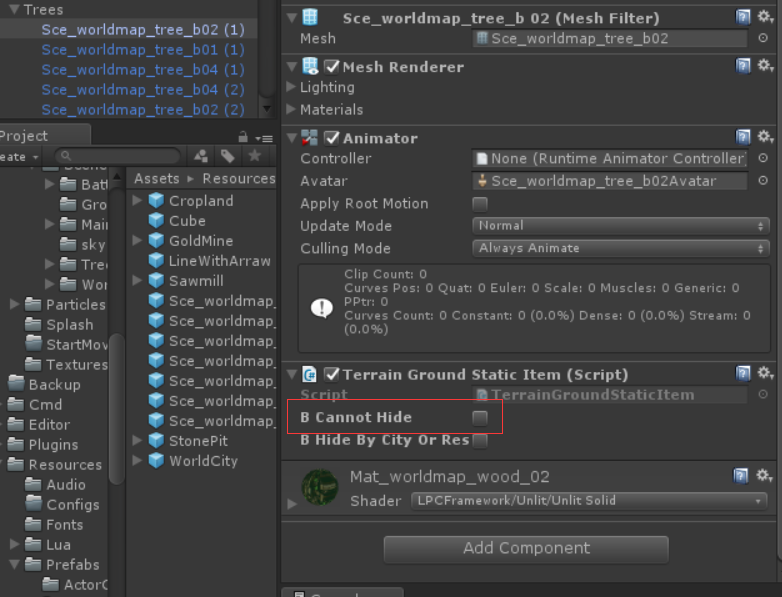
对于资源建筑只需要指定shadow projector（显示影子）

Projector mats对应于边界projector的材质球。

Shadow projectors对应于各个级别的模型对应的影子的材质球。

1. Terrain Ground Static Item

地面不会动的树和石头需要挂static item脚本，一方面用于优化（显示和影藏），一方面用于摆城时，影藏城市所在位置的树。

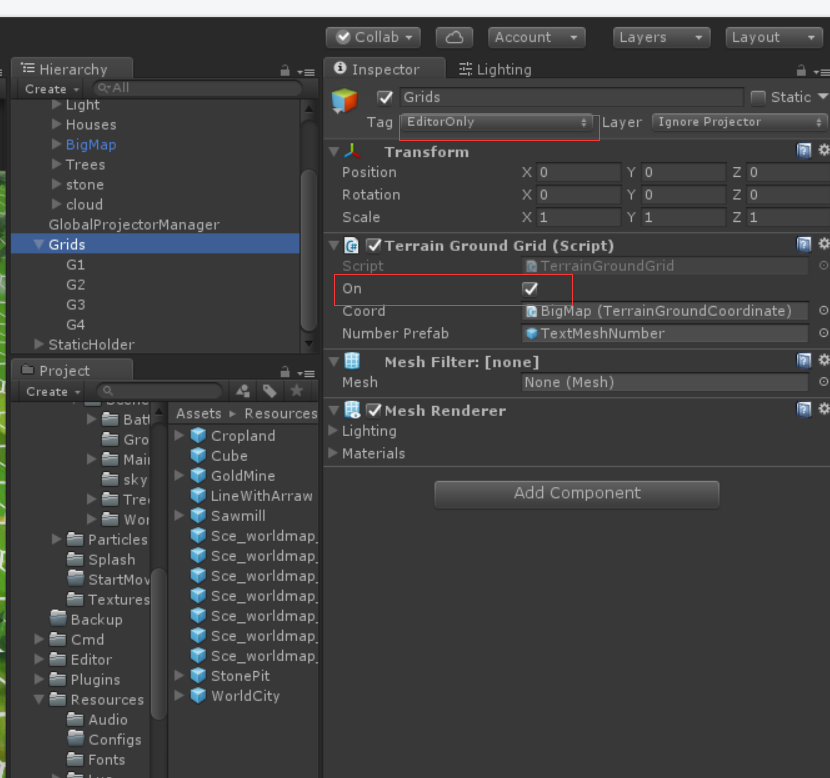


对于大石头，建议勾选cannot hide。

注意：如果用了对其地表工具，会自动添加这个脚本。

1. Terrain Grid

格子系统是给策划用的。

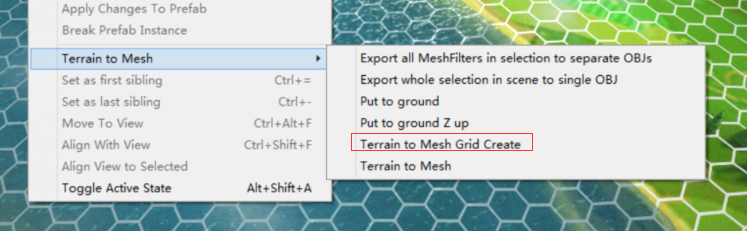


注意需要勾选On来决定是否有格子系统。另外建议tag为EditorOnly避免打包打进游戏。

**Tips1: 如果需要发布版本给策划同学用，tag不要选择EditorOnly，否则这些prefab没法被打包！！！**

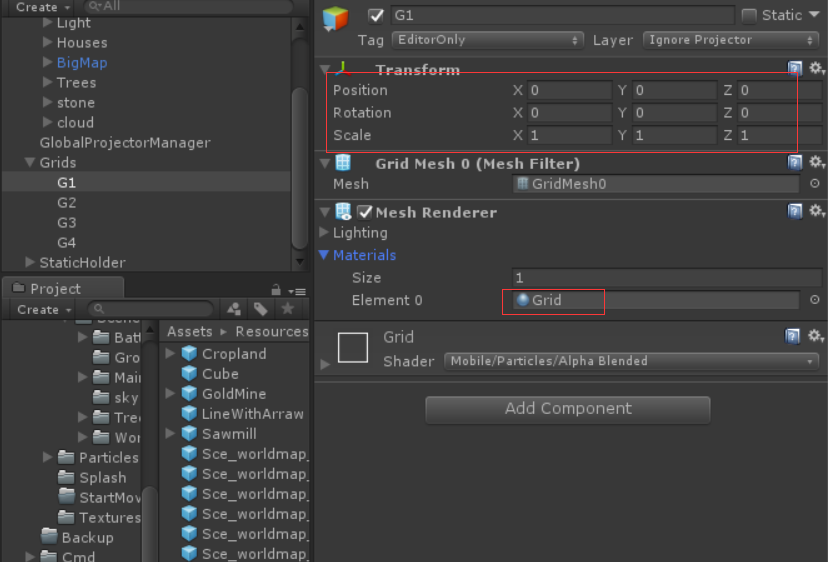
**Tips2: 打开世界地图场景，运行，再继续下面的步骤！！！**

1. 首先生成格子mesh。如图



拖入Terrain Ground Coordinate就可以生成一套格子mesh。

1. 将生成的格子mesh拖入到Grid的子物体。并指定材质球：



1. 将coordinate和text mesh prefab拖入：



1. 加入空的mesh filter和mesh renderer

如上图。

用于高亮一个格子。

这样地表格子就可以使用了。

1. 关于优化：
   1. 碰撞检测。

代码里还留有根据高度图来计算位置的代码（注释掉了）。如果以后优化collider可以作为参考。

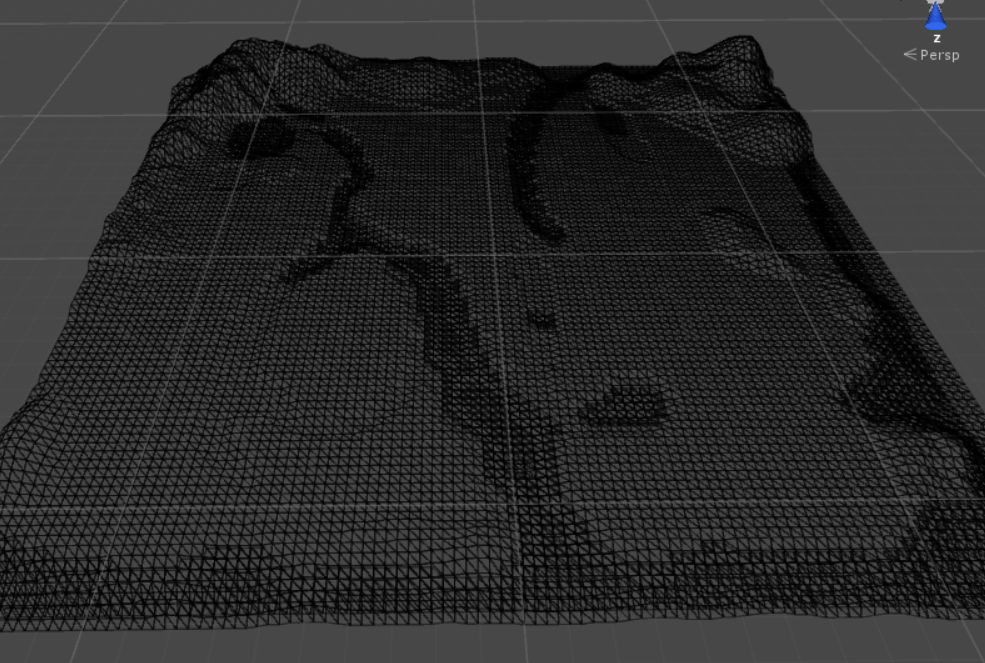
最初的系统是通过高度图来计算位置的。因为美术后面不再使用terrain转mesh，就废弃了。重新使用这套系统可能可以优化目前的collider碰撞检测。

（我建议先尝试给一个减过面的mesh来优化。而不要用这一套。）

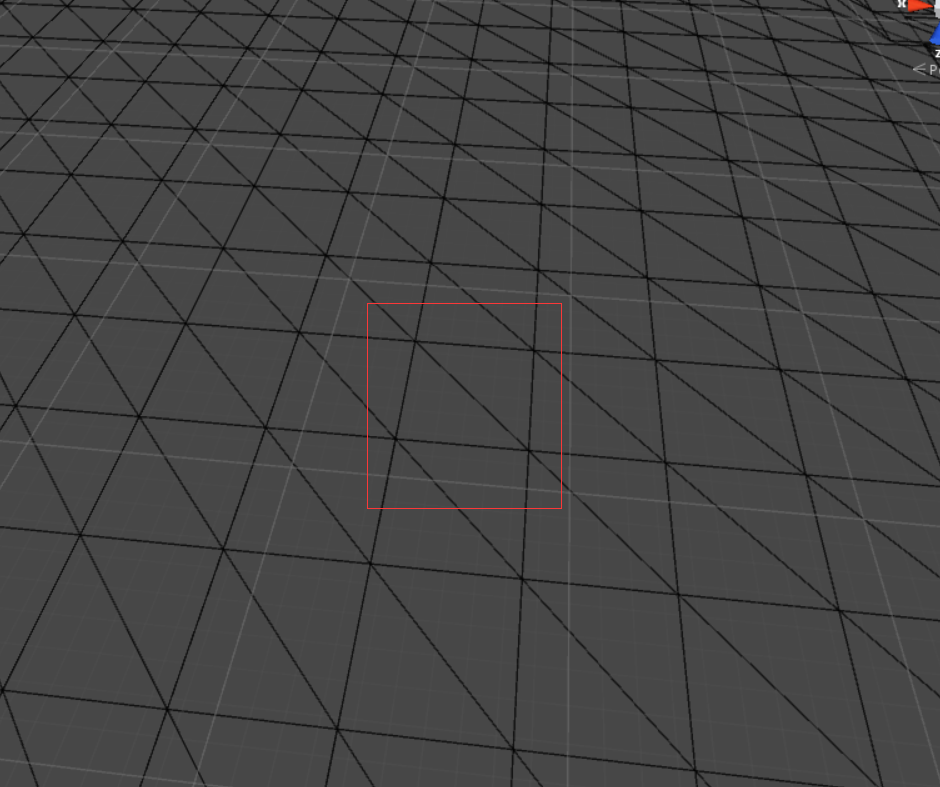
1. 重新计算高度图。

对于任何mesh。指定一个贴图大小。比如512x512。总能根据坐标系统计算出每个像素对应的高度。只要分辨率够，一定能足够好的近似表示出mesh。

以平面为例。相当于把任意mesh用这样的方式近似表示：

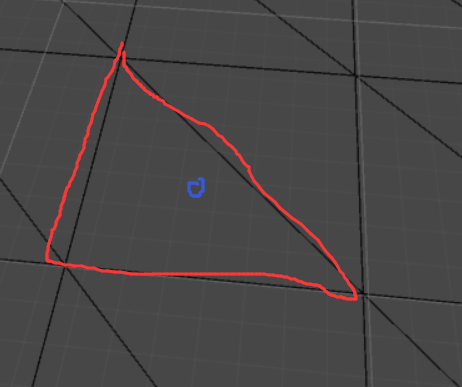


它的好处在于，可以快速的根据最上面提到的uv坐标，给出地面上对应的正方形。左边上边的顶点的x,z，分别是u\*512,v\*512向下取整，右边，下边的顶点的x,z分别是u\*512,v\*512向下取整。然后根据x,z以及高度图，就知道对应的y值。



1. 知道正方形，求位置和法线：

首先判断属于哪个三角形：



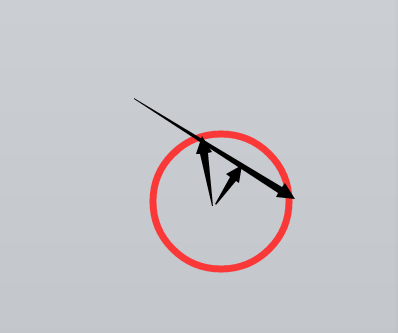
然后根据传说中的“双线性”什么的计算。即可得到。相应的代码在注释掉的部分。

1. 射线碰撞检测。

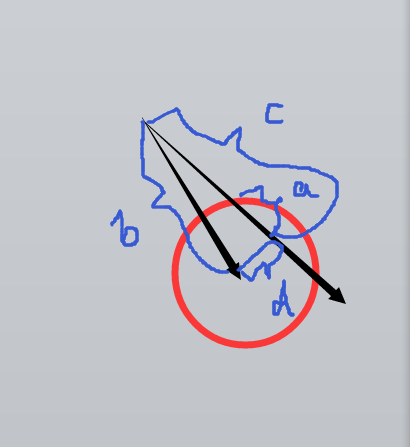
平面的碰撞检测应该不是问题，就不细说了。曲面的如下：

为了计算摄像机正对位置，或者鼠标点击位置，需要知道射线碰撞检测。它分为两步：

球碰撞检测：



先计算球心到射线的距离d，然后用半径的平方减去这个距离的平方（如果小于0表示没有碰撞到。）计算下图中的a。知道d，以及射线出发点到球心的距离b，就可以计算出c。然后c-a就是碰撞点。



第一步：先用平均半径，球碰撞计算一次位置，

第二步：在用该位置的半径，再一次球碰撞检测计算一次位置。

根据我的测试，两次计算对于地面没有剧烈起伏（山顶之类），已经能得到正确结果。

相应的代码在注释掉的部分。

* 1. Projector

目前的优化感觉和面片是一个效果。

如果需要贴着地表的边界线，可以根据边界对应的uv坐标，计算出地面位置，然后画line render或者自动生成一个mesh。

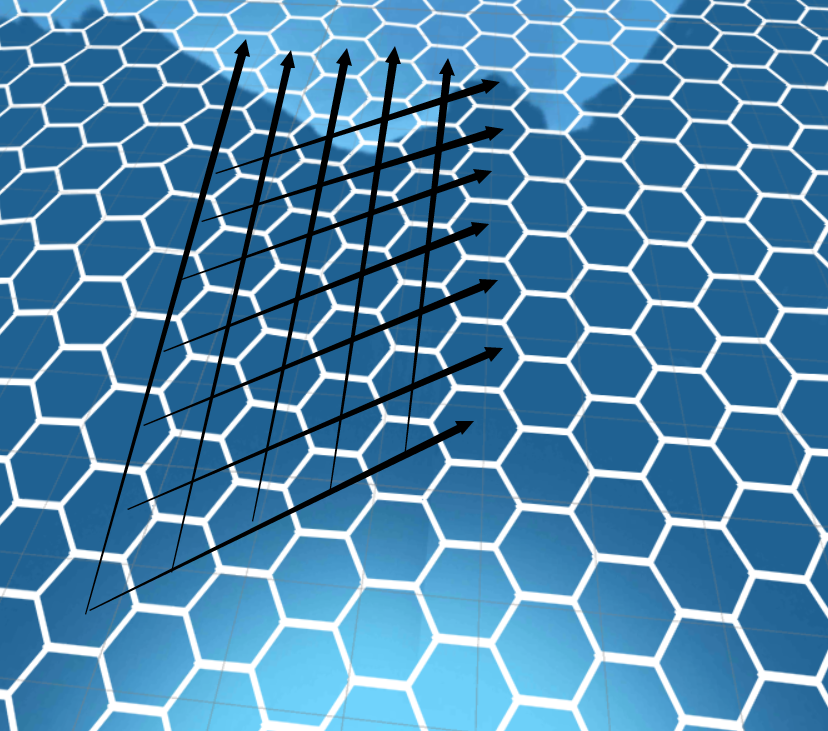
后面这一步有相应的接口，前面的这一步比较麻烦。

建议的计算方式：（未测试，只是给一个思路）

1. 把OddQ坐标对应到斜四边形坐标。

需要注意，对于x为奇数偶数，对应规则不同。

如下图：



（2）

这样就变成了在四边形格子里找边界的问题。

比如

01110

00100

01110

这样的填充

只需要做一些移位操作就能找到边界。

比如右边界：

000011110001100

要得到

000000010000100

可以经过先反

111100001110011

与自己的反的右移

000011110001100

000001111000110

111100001110011

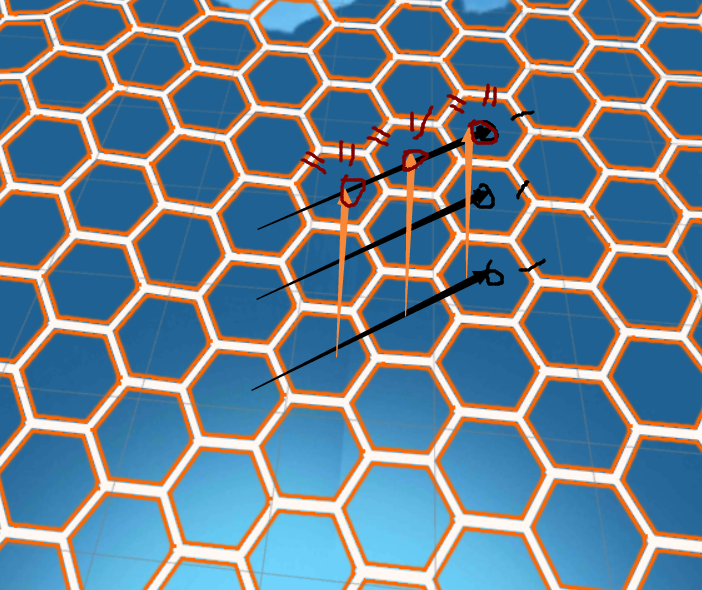
&

000001111000110

求且得到：

000000001000010

计算出来右边界后，画它们的右边。计算上边界，画上面两个边：



依此类推画左边，下边。最终画出边界。