已有的方块已经全部做成预制件存放在prefabs/Block中

|  |  |
| --- | --- |
| Grass | 对应8中不同样式的草地方块 |
| Black | 黑块 |
| Gold | 金块 |
| Jump | 跳台 |
| Up | 气凝胶块 |
| Vine | 藤蔓 |
| Wood | 木块 |

|  |  |
| --- | --- |
| GlobalController | 存储地图的初始信息，用二维数组存储地图高度、藤蔓方块位置、木块位置、黑块位置、金块位置、跳台位置，起点终点。  实现关卡切换和UI交互 |
| LevelStart3  LevelStart4  诸如此类  LevelStartN | 这个是GlobalController之后才写的，目的是向GlobalController中写入不同的地图信息（当前关卡的地图信息），因为GlobalController默认只有第一关的地图信息 |
| EnergyController | 该物件自主场景mainscene被创建，即使切换了关卡也会一直保留下来，保存了能量条的全部信息，与黑块和金块的交互，都会反映到该类中 |
|  |  |

主要功能模块：

|  |  |
| --- | --- |
| 拖拽模块 | 主要有DragBlock,HeadBlock,BodyBlock三个类，在3d场景中，分别对应了三个物体，其中DragBlock对应的是空物体  以草地方块为例  计算机生成了可选文字: 卜9.'assl base 卜仁r,rr,  grass1对应的是上面的草地块，对应的类是HeadBlock，base对应的是下面红色的土壤方块，对应的类是BodyBlock。  在初始状态下，土壤方块在竖直方向的大小（scale.y)是0  通过点击HeadBlock部分或者是BodyBlock部分，实现二者同时被拖拽，即BodyBlock随着拖拽而沿y轴向上放大，HeadBlock沿y轴上升。  DragBlock中则存储了该方块的坐标信息，即该方块是第几行第几列，以及可以达到的最大高度MaxHeight是多少，该坐标信息会对应到GlobalController的map数组下标，创建新关卡的时候必须手动设置每一个方块的坐标（map数组下标)    需要注意的是：拖拽模块对于方块轴心点的位置和相对于父物体DragBlock的位置，以及碰撞盒的大小极其敏感，任何参数上的细微变化都有可能导致诸如对不齐之类的bug    DragUpBlock,HeadUpBlock,BodyUpBlock与之类似，仅仅是用于气凝胶方块的拖拽  其他方块都是用的是DragBlock,HeadBlock,BodyBlock,木块没有赋予任何脚本 |
| 寻路模块 | 采用A\*寻路算法，这是一个经典的广泛用于游戏的寻路算法    首先，该算法会从起点开始，依次遍历当前点的 后、左、前、右 四个点并加入到待查列表OpenList中。右指的是x轴正方向，具体的方向需要在场景中查看一下每个DragBlock中的坐标为之（map数组下标)，。  每个点都会有一个估计的权值，就是该点到终点的横向距离+纵向距离。  下一个将要分析的点就是OpenList中权值最小的点，如果权值相同的有多个，就取最后加入的那个。  同时，还需要判断相邻的这个点是不是可达，比如map[i,j]<0说明不可达，或者高度差绝对值大于1说明不可达，或者当前方块是跳台高度差绝对值小于等于2那么可达    输入的是一个map数组以及起点终点，其中map数组是会随着玩家拖动滑块而改变的地形信息。  通过该算法最后会得到一个名为Path的链表，其中存了最短路径的一系列坐标节点。    至此其实寻路的路径求解部分就已经结束了。而另一部分就是播放的时候根据map数组下标获得Unity3D场景中的空间坐标，采用的是MapController中的getV3方法    获得每个节点的坐标后，使用一个小智写的移动函数，即可实现任意两个坐标点间的移动，如果有高度差就跳跃，没有高度差就平移    在移动过程中，依然可以获得当前所在砖块是什么砖块，由此触发一些函数，比如黑块，金块都是这样做的 |
| 跳跃模块 | 小智写的部分，我没有细看  我所了解的是，用物理引擎编写，但并未使用碰撞检测，平移滚动如果遇到障碍会失控 |
|  |  |