光，影子，朝圣者 优化方向内容

原代码重构

1. **LightExtension**现在的检查逻辑是直接实时获取光照组件的方向，这将导致在光照旋转/移动动画过程也会影响游戏场景(这一点在最后一关影响很大)，需要将逻辑计算与实际动画分离，即将**GlobalLight**的**LightDir**数据单独分离出来，在给予旋转/移动时直接一次性修改向量值，随后再由动画修改光照组件的实际位置，分离逻辑与表现，这个修改对**后续游戏设计很重要**

   注：曾测试过这样的修改会**导致两处地块的LightExtension的OnLighted和OnDarken同时触发**，需要处理好先后顺序（前一个的OnDraken => 后一个的OnLighted），之前因为太懒又改回去了(

2. **效果表现器**的独立，现在的初始动画(依次升起物块)以及玩家点击效果(白圈)被堆在了**StartEffect**中，然后这个脚本被挂载到了Map上，不难看出这是托极不合理的屎山(，需要把这些效果的管理独立出来放到一个独立的Manager里更合适

   StartEffect本身也有点史，只单独的对特例进行了特殊的if处理，很臭，如果想改下也可以(

3. 屎山重灾区**主菜单**, **MainMenuManager**非常的史，虽然能跑()，后续的设计可能涉及切换章节(更换一个新时钟)，调整一章的关卡数量(修改每次旋转的角度)，当然，如果不改应该影响也不大

4. 现在转场时，为了保证无缝衔接，主播干脆直接把Canvas的颜色修改成了游戏场景的同色，不过考虑到后续场景颜色可能要微调，最好还是给每个场景设置一个背景色变量，并在转场时将该颜色传给canvas，(是不是设置一个ScriptableObject专门存储每个场景的相关信息会更好呢O.o)

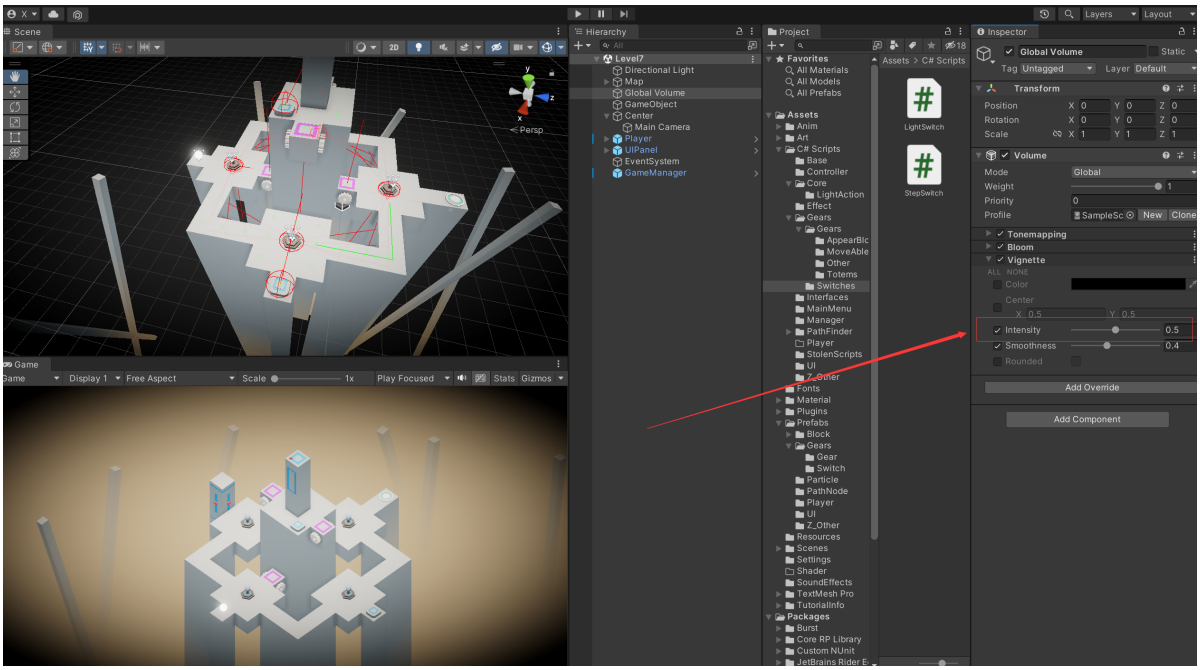
5. 有个小bug，游戏里其实是修改了鼠标样式的，但是不知道为什么编辑器里可以用，到了实际游戏里样式就没变了

6. 可以的话直接把之前写好的大移动板的代码搬进来，省的一个个写delay(

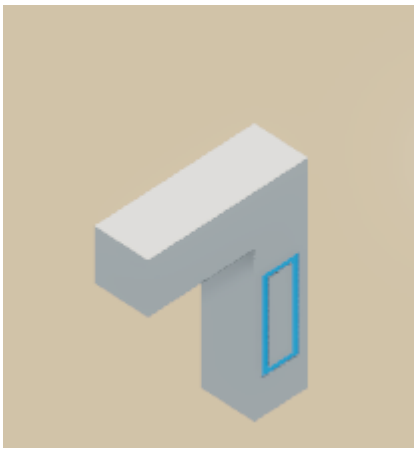
新功能设计

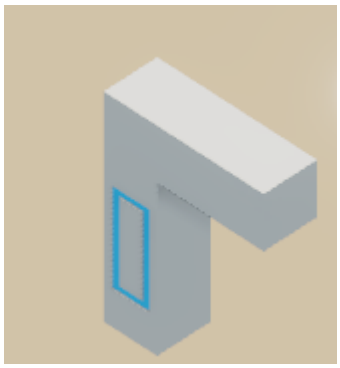
在接下来的章节中，计划设计以下的新功能/设计

1. 阻滞移动 当玩家以任何方式导致自身处于与**自身状态相反** (这个概念后续会解释，暂时可以理解为玩家因旋转光线等原因被迫站到了阴影下)的地块时，禁用玩家的移动操作，并将界面状态以一个动画形式修改以提示玩家位置异常，目前的设计上，可以考虑将**GlobalVolumn的Vignette**的Intensity修改成0.4~0.5的数值，并以微小的数据不断跳动，形成类似于Apex倒地快嘎掉的感觉

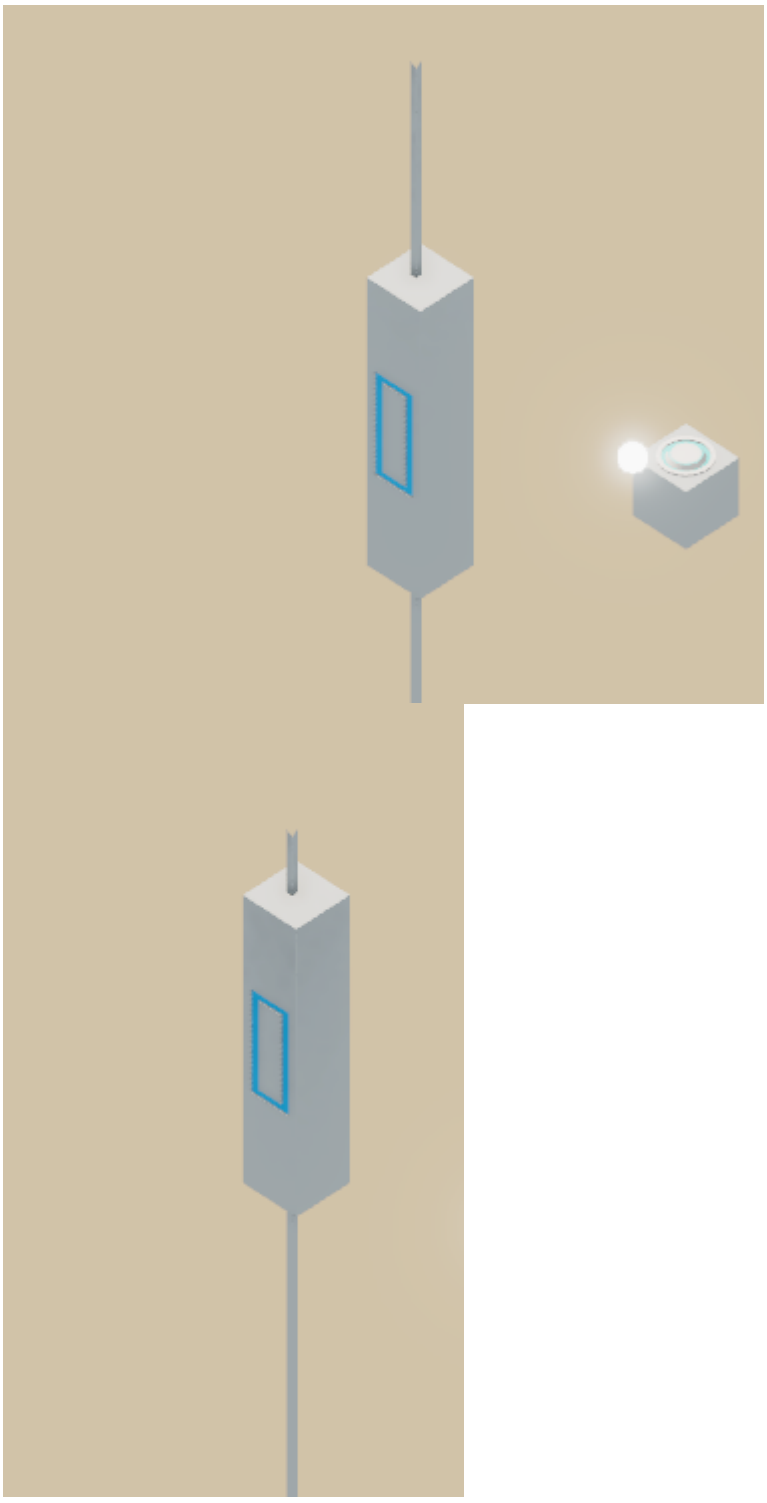


2. 旋转机关 以机关形式出现，单击或以其它形式(按钮等)触发时它时旋转90 / 180 度， 它应当有 Loop 和 PingPong 两种模式 (参考光线旋转图腾)， 由于玩家在移动到一个新地块时会直接认其做父物体，应该只要旋转自身，不用考虑如何带动玩家

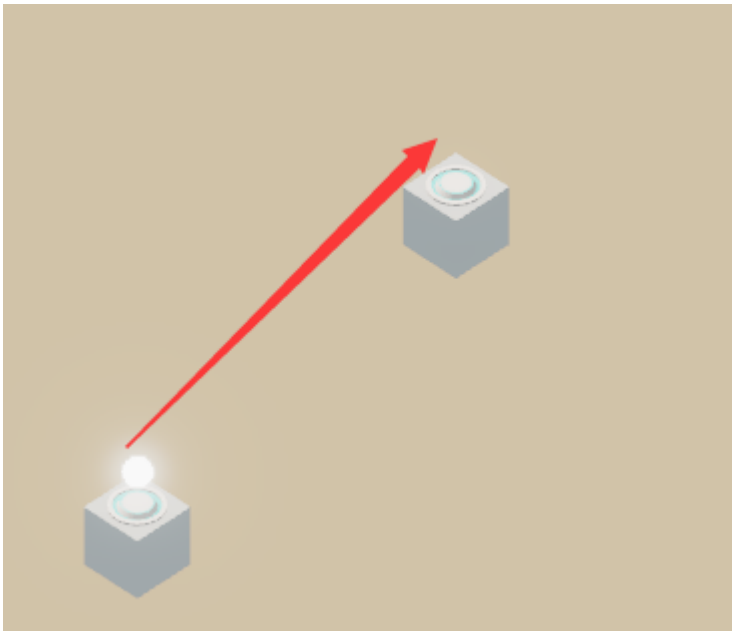




3. 可拖动地块 允许玩家在既定轨道的物块，操作方式是点击操控位置并按住向某方向拖动(见纪念碑谷的移动块)



4. 传送点 当玩家进入后可以传送到对应的另一处，原本的移动块在垂直方向的移动极易出现卡关问题，所以改成这种东西用来方便垂直移动



5. 暗形态 游戏的下一项核心内容，包括以下内容

   暗庭

   一个特殊地块，此地块同属两种类型的地块(光和暗)，总是会被允许纳入寻路路径中

   当玩家进入暗庭后，强制终止玩家移动路径，进行一个过程，这个过程将涵盖

   - 将背景修改为更暗的颜色
   - 调整玩家的色彩(最好是黑色，但我目前搞不出发荧光的黑色)
   - 修改玩家的形态为相反形态，这个形态下玩家**只可通过暗色地块**

在原有的代码中，已具有**GameManager.Instance.currentPlayerState**存储了玩家的位置，而地块的可达判定为

```
public override bool ReachAble(LightState inputState)
{
    return inputState == lightState; //判定当前玩家状态与地块状态是否一致
}
```

只需稍作修改，在玩家进入时将currentPlayerState改成暗色，即可修改玩家状态

在暗形态下，玩家进入光照地块时同样进入 **阻滞移动** 状态

当玩家以暗形态进入一个暗庭，玩家回归正常状态

移动端

为了移植到移动端，游戏设计中尽量避开了键盘操作，并且对所有的鼠标操作都分离出了单独的函数

在完成游戏开发后，准备将其移植到手机端上(横屏)，因此代码记得留好接口