CottonPuzzle

推箱解谜Mock-up

用时：

9小时左右（周日上午10-12，下午4-7，周一凌晨12-4）

目标达成：

SpriteRenderer游戏逻辑，
UGUI游戏UI，
背景音乐及暂停/播放，
后退/重新开始按钮的Animator动画

考虑到实际生产环境中方便迭代关卡等因素，决定使用JSON文件读取关卡配置。

文件结构

DataParser.cs 读取并解析JSON字符串，将关卡配置存放于Serializable Object中。
GridManager.cs 根据关卡配置在场景中实时放置关卡元素，可分别绘制Grid基本层（存放基本棋盘格，不会移动的终点和箭头）与Token棋子层（存放可点击和移动的棋子）。
GameManager.cs 为核心玩法功能脚本，通过与GridManager及各个方格交互控制游戏流程。
UIManager.cs 为UI控制脚本，所有的按钮事件都与其绑定，由其分发。
Tile.cs 为格子基类。
InteractableTile.cs 继承自Tile，额外支持棋子的移动等功能。

具体实现

JSON关卡

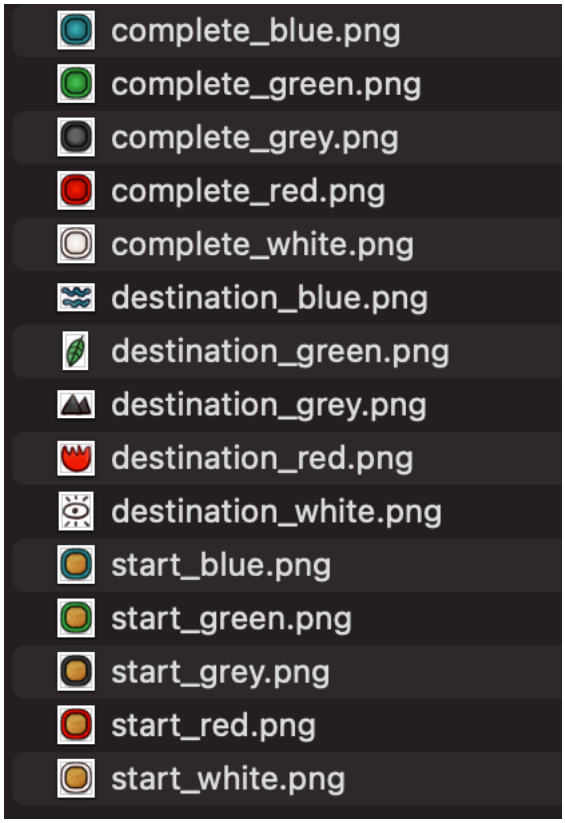
```
"items": [
    {"type": "destination", "x": 3, "y": 2, "family": "white"},
    {"type": "destination", "x": 4, "y": 3, "family": "red"},
    {"type": "start", "x": 3, "y": 1, "family": "white", "direction": "up"},
    {"type": "start", "x": 4, "y": 2, "family": "red", "direction": "up"}
]
```

只需要注明坐标，地形种类，颜色和方向即可表示一个最简单的关卡

关卡绘制

```
var posX = offsetX + x * defaultTile.transform.localScale.x;
var posY = offsetY + y * defaultTile.transform.localScale.y;
var currentTile = Instantiate(defaultTile, new Vector3(posX, posY), Quaternion.identity);
```

可在编辑器中调整绘制偏移量，并根据格子大小调节绘制。保证了之后如果有版本地图不再是8x7的标准大小，也无需重写代码。



重新规范命名了图片素材，不需要拖拽任何一张Sprite，程序只需要使用JSON中提供的信息就可以自动获取并绑定所需的Sprite。

箱子移动

```
foreach (KeyValuePair<Vector2, InteractableTile> token in gridManager.tokens)
        {
            if (token.Value.pos == tile.pos && token.Value.family != tile.family) {
                if (!TileMove(token.Value, tile.direction)) {
                    tile.pos = tile.pos – DirectionVec.directionVec[tile.direction];
                    return; }
            }
        }
```

在移动箱子遇到前方箱子阻挡时，使用递归优先处理最前方的箱子；如果最前方的箱子触碰到地图边界，则会依次撤销每个箱子的移动，从而维持整个关卡数据的合法性。

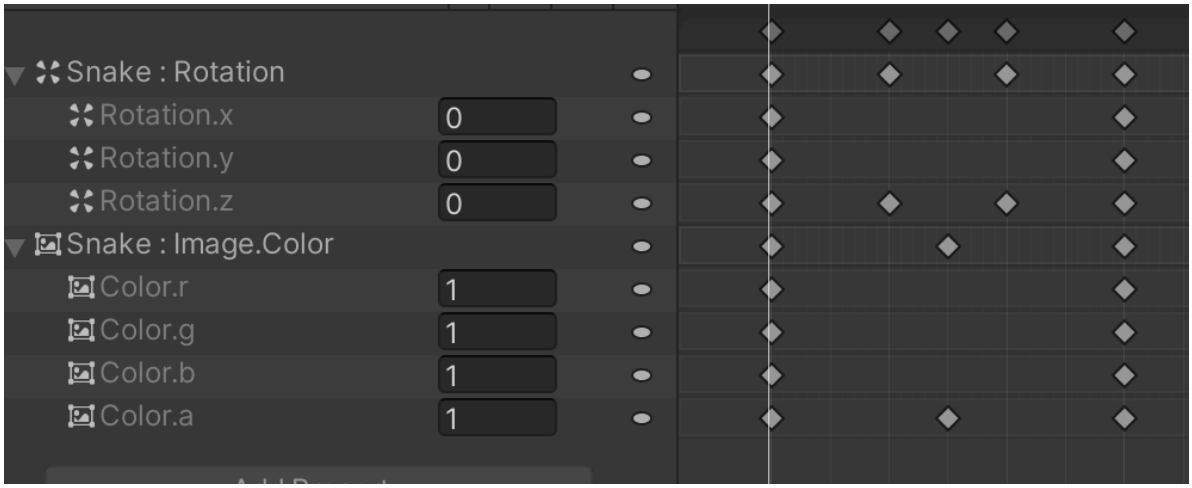
后退一步

```
List<InteractableTileData> currentState = new List<InteractableTileData>();
        foreach (KeyValuePair<Vector2, InteractableTile> item in gridManager.tokens) {
            currentState.Add(item.Value.StateFactory());
        }
        timeMachine.Add(currentState);
```

借鉴了一些工厂模式的灵感，在InteractableTile中建立一个InteractableTileData工厂，每次成功移动后将当前所有棋子的状态保存在一个List中。所有的List由timeMachine管理，本质上是一个栈，玩家操作后新状态入栈，每次点击回退则利用栈顶所存的状态，调用GridManager重绘token层。

按钮动画

重新开始和后退一步按钮都用Animator+Animation制作了简单的动态效果，不过因为隔离点开始赶人了没有抠细节。



简单的UI关键帧动画，给蛇加一点动态效果。

不足之处

动画的曲线和过渡没有微调，例如后退按钮的蛇动画其实可以使用两张Sprite交叉过渡，看起来应该会连贯很多；箱子的移动也可以加上动画，其他按钮可以增加一点反馈。

本来在随机选关卡的时候想了一段既可以只生成一次随机数，保证随机性的同时也能稳定排除掉已经通过的关卡的代码。试了几下没搞对，考虑到时间性价比还是使用了最基本的“一直随机到不重复为止”方法。

主动移动和被动移动箱子两个方法有很大的重合部分，我只是简单的用不同的参数重载了，应该有更好的复用方法，面向对象得多看看。

背景音乐静音试了几种方法都没反应，最后用了全局的AudioListener。如果自己写一个AudioManager应该会方便一些。