**1.状态控制方案**

**游戏状态机，** 控制游戏的整体流程 StateMachine.cs

任何游戏，都会经历 游戏开始 -> 游戏运行 ->游戏结束 阶段

传统的游戏流程控制方案是 if..else... 或者 switch

例子： int game\_State = 0;

if(game\_State == 0)

{

需要执行的一大串脚本....

}

else if(game\_State == 1){}

else if(game\_State == 2){}

缺点：当游戏状态过多的时候， if..else需要写很长一串，且不好管理状态切换间的事件

现方案：每个状态实体挂一个状态机脚本。 将一个个状态分为若干子类，每个子类里有进出状态的事件。 enter/ Excute/ Exit;

如 ：

game\_State == 0 --------> GameBeginState.cs

game\_State == 1 --------> GameRunState.cs

game\_State == 2 --------> GameEndState.cs

状态间的一大串脚本，放入状态脚本中执行

1. **游戏的数据存储逻辑**

**主要使用PlayerPrefabs进行数据存储/读取**

**GameSaveManager.Save<int>(dataType, 1);**

**GameSaveManager.Load<int>(dataType);**

1. **入门使用**

**所有游戏主要核心逻辑从此处进入。**

**红框以下的代码只是底层方法往上传，不调用时候在游戏中不起作用。**

**状态内的脚本都可以删掉重写。**



